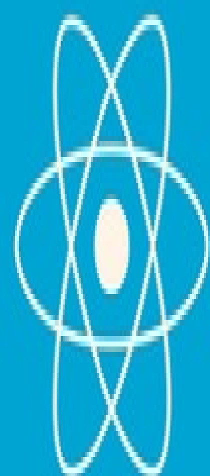


React Native

Table of Contents

React Native Training



The videos are here (YouTube, YouKu (中文)!

Todo

React+Redux DEMO for React Native Training

69

Go Home	02:07
Go work	00:07:28
Go party	completed
Go swimming	03:07
Go riding	02:07
Go walking	01:07
Go running	02:07

Home Completed Book

Go Home	7 mins ago
Go work	3 hours ago
Go party	3 hours ago
Go swimming	7 mins ago
Go riding	52 mins ago
Go walking	7 mins ago
Go running	7 mins ago

Home Completed Book

Please leave a message or twitter @unbug for further enquiries. Any help will be appreciated :)

Created by @unbug:

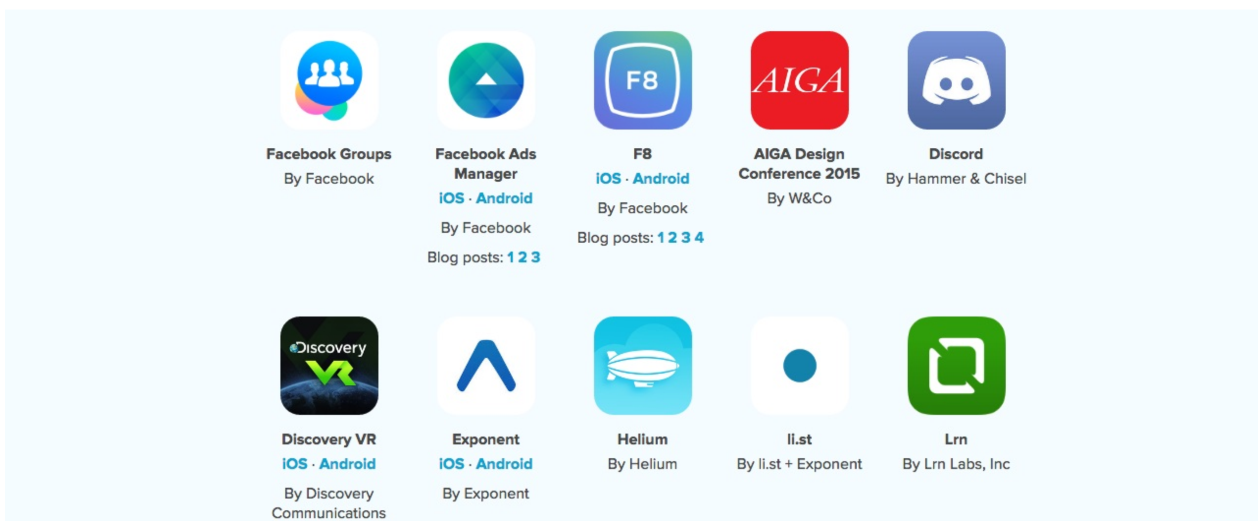
- MIHTool - iOS Web Debugger Pro: MIHTool helps Front-End Engineers to debug and optimize their webpages on iPad and iPhone.
- Codelf - 变量命名神器: Organize your GitHub stars and repositories.Search over projects from GitHub to find real-world usage variable names.
- js-middleware: Powerful Javascript Middleware Pattern implementation, apply middleweares to any object. A painless solution to make codes as scalable and maintainable as ReduxJS and ExpressJS.
- SAY NO TO SUICIDE PUBLIC LICENSE: We've lost so many genius developers, who committed suicide, such as Aaron Hillel Swartz (November 8, 1986 – January 11, 2013). As a developer, the community needs you, the world needs you, please keep yourself alive.

1 First Look

Introducing React Native

> What we really want is the user experience of the native mobile platforms, combined with the developer experience we have when building with React on the web. With a bit of work, we can make it so the exact same React that's on GitHub can power truly native mobile applications. The only difference in the mobile environment is that instead of running React in the browser and rendering to divs and spans, we run it an embedded instance of JavaScriptCore inside our apps and render to higher-level platform-specific components. It's worth noting that we're not chasing **“write once, run anywhere.”** Different platforms have different looks, feels, and capabilities, and as such, we should still be developing discrete apps for each platform, but the same set of engineers should be able to build applications for whatever platform they choose, without needing to learn a fundamentally different set of technologies for each. We call this approach **“learn once, write anywhere.”**

Showcase



1.1 Building an app in 5 minutes

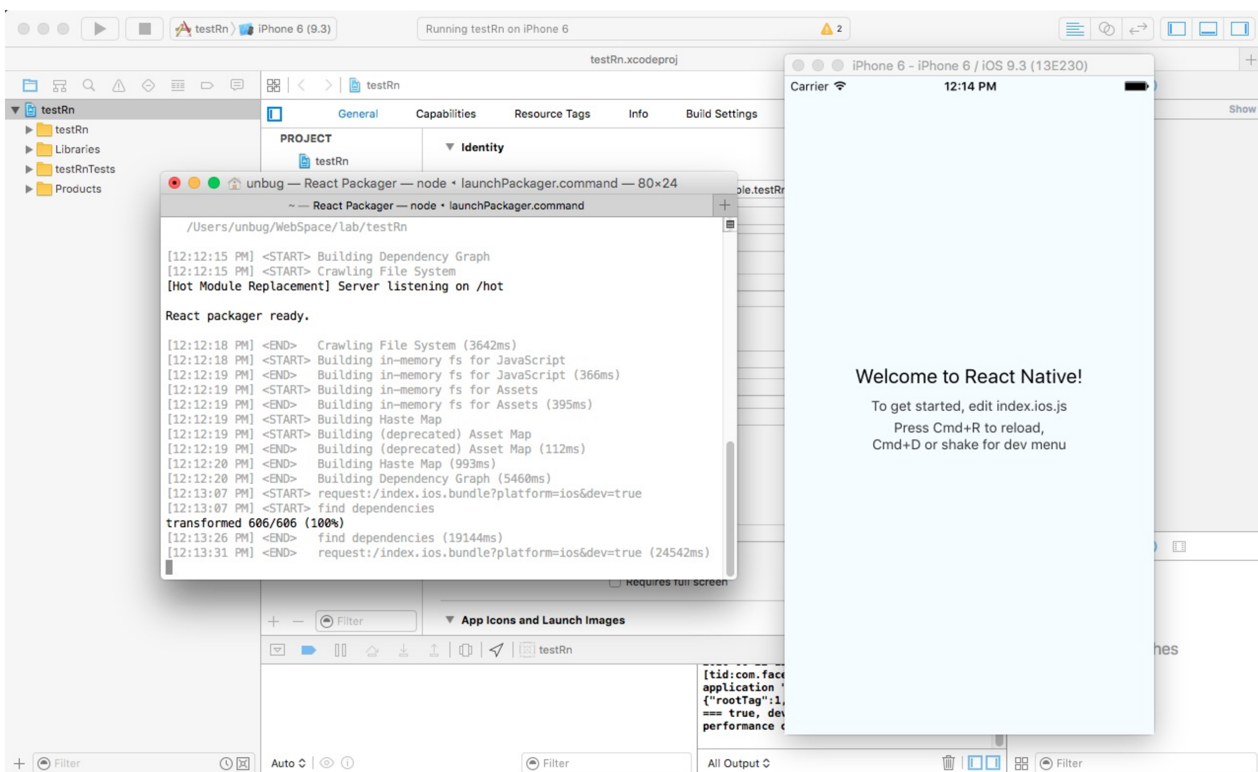
1. Requirement follow Getting Started
2. Generate a new React Native project

```
react-native init testRn
```

3. Build & run project

```
react-native run-ios
```

or open `testRn/ios/testRn.xcodeproj` and build with XCode's play button

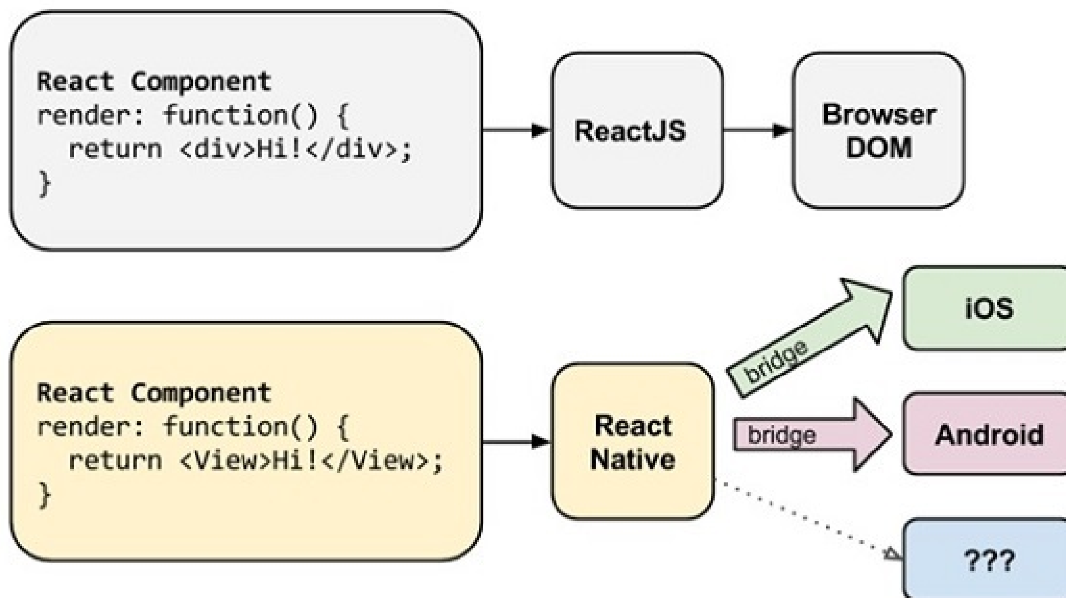


or if the app already builded, start the webserver

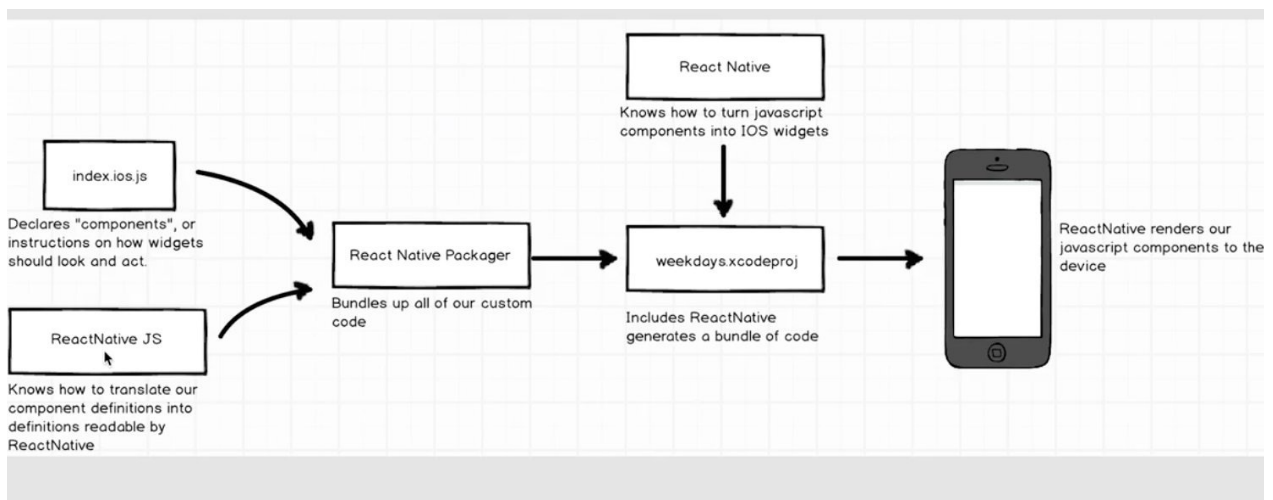
```
npm start
//or
react-native start
```

1.2 How it works

1.JavaScript bridge

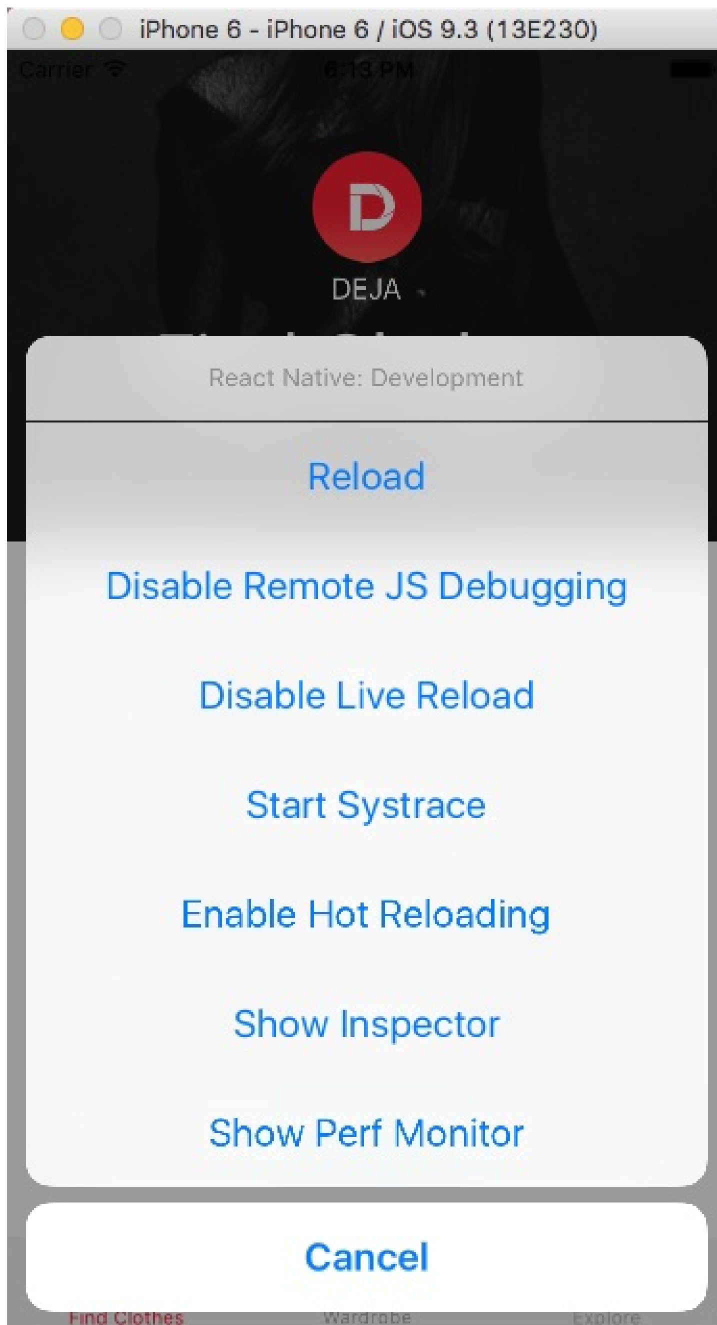


2.React Native Packager

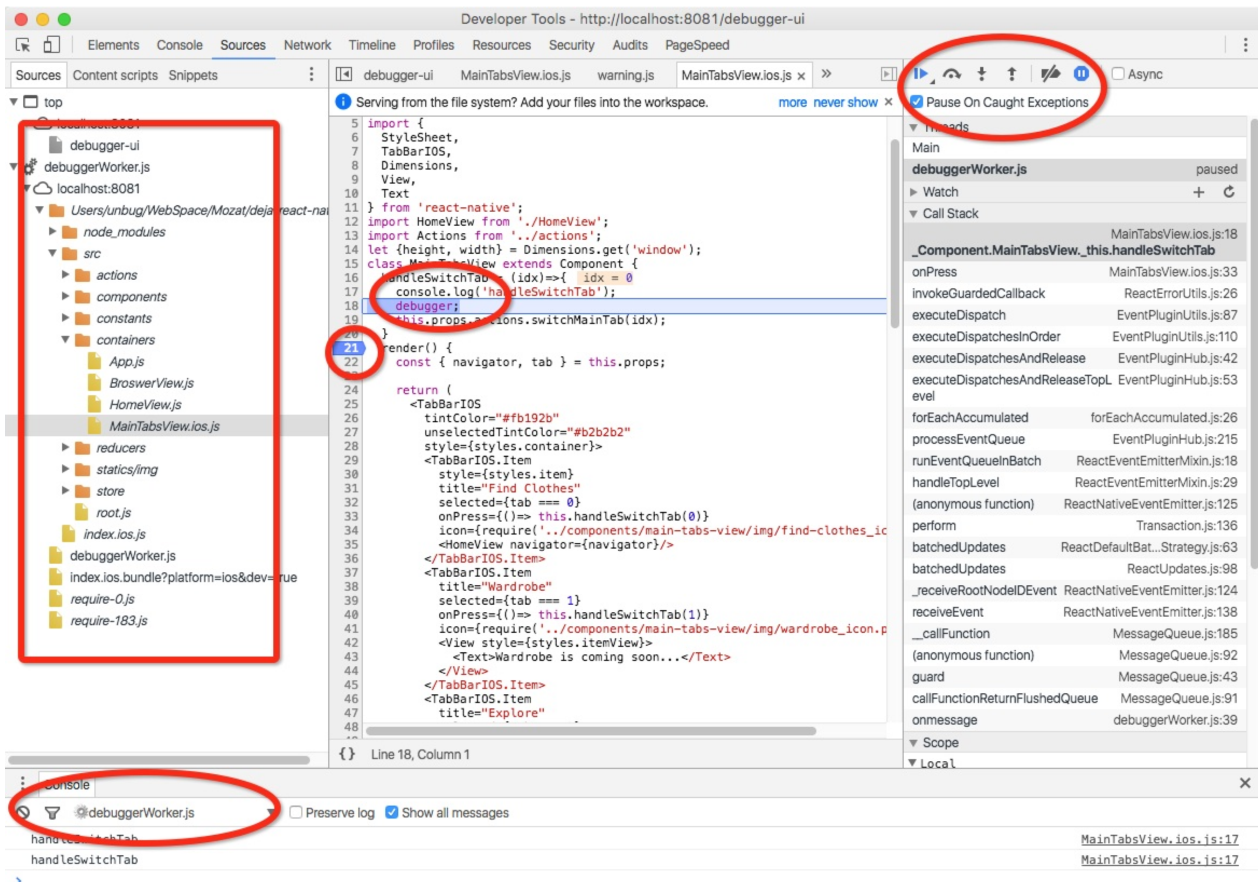


1.3 Debug tools

1.developer menu



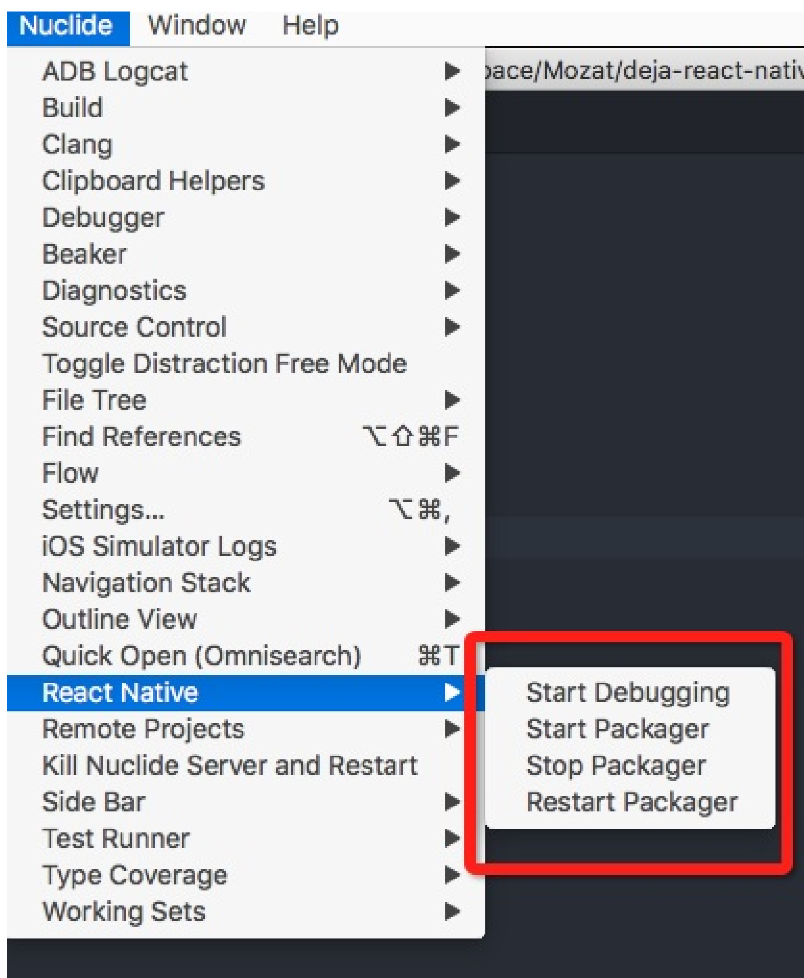
2.Chrome Devtools



3.log

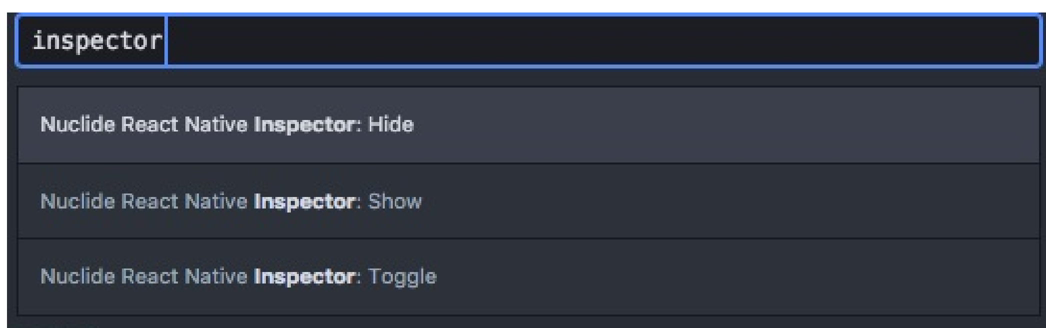
```
console.log('some text');
console.dir({a:1, b:2, c:3});
debugger;//breaking point
```

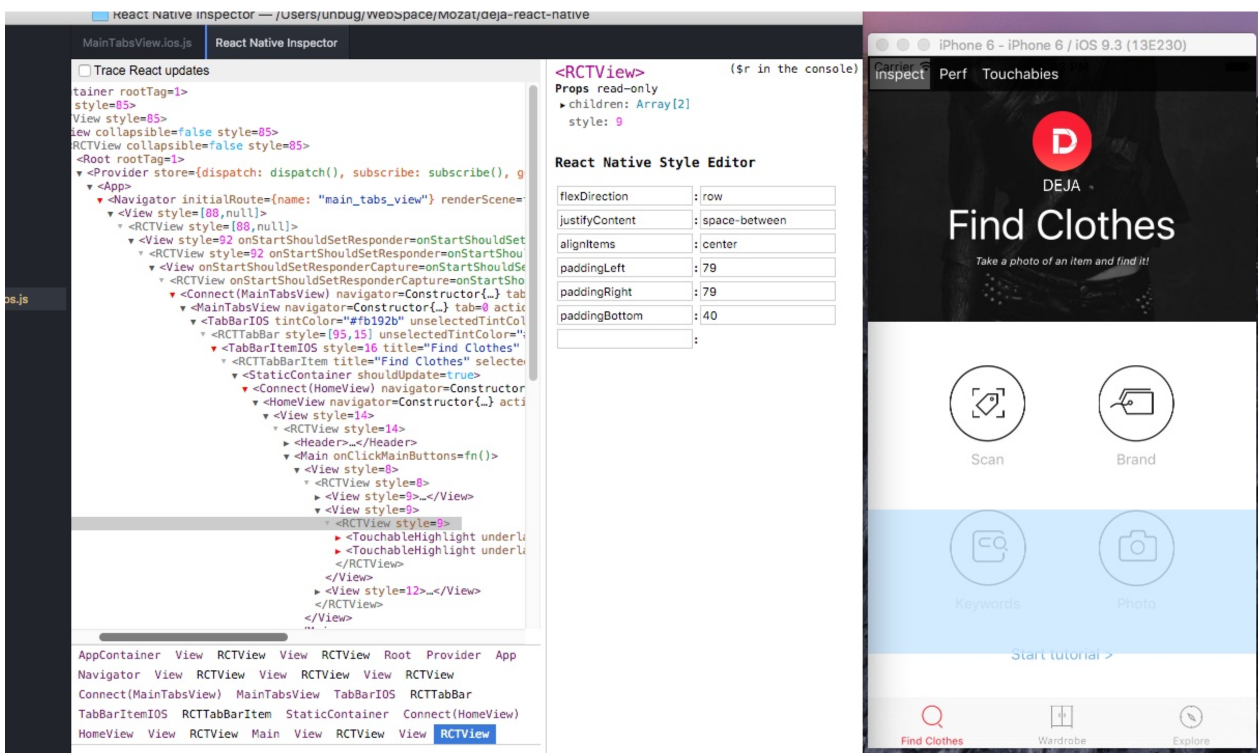
4.Atom & nuclide



5.inspect

Open Atom Command Palette package with `cmd-shift-p` and search "inspector", then click "Nuclide React Native Inspector:Show"





6. Real device

6.1 Deploy to real device `project_name/ios/project_name/AppDelegate.m`

```
  //jsCodeLocation = [NSURL URLWithString:@"http://localhost:8081/index.ios.bundle?platform=ios&dev=true"];

  /**
   * OPTION 2
   * Load from pre-bundled file on disk. The static bundle is automatically
   * generated by the "Bundle React Native code and images" build step when
   * running the project on an actual device or running the project on the
   * simulator in the "Release" build configuration.
   */

   jsCodeLocation = [[NSBundle mainBundle] URLForResource:@"main" withExtension:@"jsbundle"];
```

6.2 Debug in real device

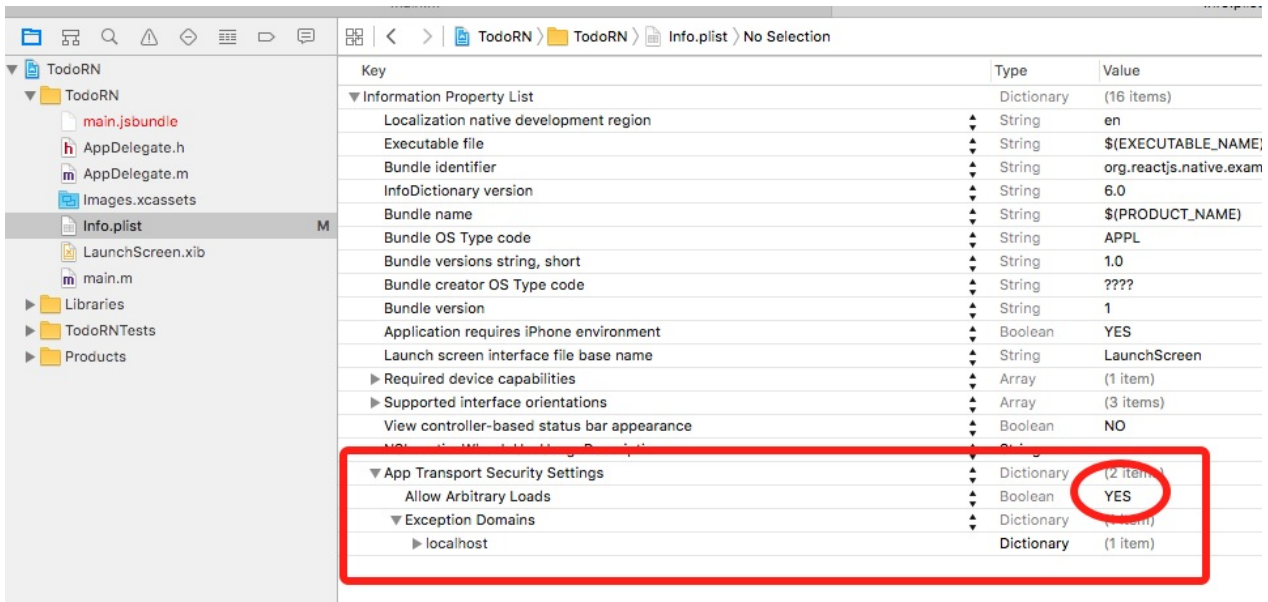
1. `project_name/ios/project_name/AppDelegate.m`

```
  jsCodeLocation = [NSURL URLWithString:@"http://172.28.0.230:8081/index.ios.bundle?platform=ios&dev=true"];
```

2. `node_modules/react-native/Libraries/WebSocket/RCTWebSocketExecutor.m`

```
  if (!_url) {
    NSUserDefaults *standardDefaults = [NSUserDefaults standardUserDefaults];
    NSInteger port = [standardDefaults integerForKey:@"websocket-executor-port"] ?: 8081;
    NSString *URLString = [NSString stringWithFormat:@"http://172.28.0.230:%zd/debugger-proxy?role=client", port];
    _url = [RCTConvert NSURL:URLString];
  }
```

3.



1.4 DOCs & APIs

- ReactJS
- React Native
- Nuclide

1.5 Resources

- React Native: Bringing modern web techniques to mobile
- React Native通信机制详解
- React Native 调研报告
- React Native概述：背景、规划和风险
- JavaScriptCore
- React Native iOS 真机调试

2 Components

1.MyComponent.js

```
//define component
class MyComponent extends React.Component {
  render() {
    return <Text>My component!</Text>;
  }
}
//export component
export default MyComponent;
```

2.Main.js

```
//import component
import MyComponent from './MyComponent';
class Main extends React.Component {
  render() {
    //use component
    return <MyComponent>;
  }
}
```

3.AppRegistry

```
AppRegistry.registerComponent('MyApp', () => Main);
```

2.1 Render & JSX

```
..
...
render() {
  const txt = 'Hello';
  function say(name){
    return 'I am '+name;
  }
  return (
    <View>
      <Text>This is a title!</Text>
      <Text>{txt}</Text>
      <View>
        <Text>{say('React')}</Text>
      </View>
    </View>
  );
}
..
...
```

2.2 View, Text, Image, etc

1. Core Components

```
..
...
import {
  StyleSheet,
  Text,
  View,
  Image
} from 'react-native';

class Main extends Component {
  render() {
    return (
      <View>
        <Image source={require('./img/bg.png')}>
          <Image source={require('./img/icon.png')}/>
          <Text>
            some text!
          </Text>
        </Image>
      </View>
    );
  }
}
```

2.3 Lifecyle

1. Instantiation

   1.1 The lifecycle methods that are called the first time an instance is created

   - getDefaultProps
   - **getInitialState**
   - componentWillMount
   - **render**
   - **componentDidMount**

   1.2 For all subsequent uses of that component class:

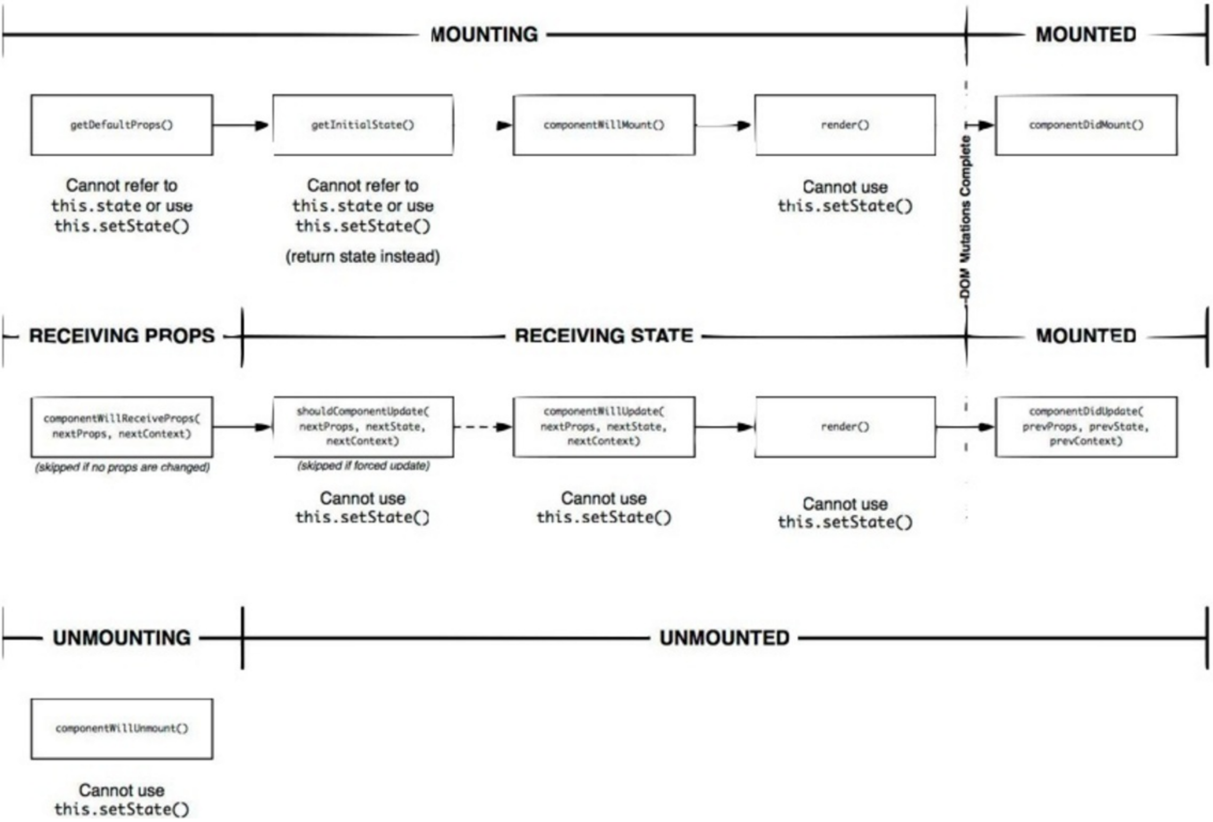
   - getInitialState
   - componentWillMount
   - render
   - componentDidMount”
2. Lifetime

   - componentWillReceiveProps
   - **shouldComponentUpdate // return true|false**

     ```
     shouldComponentUpdate(nextProps, nextState) {
     return nextProps.id !== this.props.id;
     }
     ```

   - componentWillUpdate //not called for the initial render
   - render
   - componentDidUpdate
3. Teardown & cleanup

   - componentWillUnmount


MOUNTING
MOUNTED
getDefaultProps()
getInitialState()
componentWillMount()
render()
componentDidMount()
Cannot refer to this.state or use this.setState()
Cannot refer to this.state or use this.setState()
(return state instead)
Cannot use this.setState()
DOM Mutations Complete
RECEIVING PROPS
RECEIVING STATE
MOUNTED
componentWillReceiveProps(nextProps, nextContext)
(skipped if no props are changed)
shouldComponentUpdate(nextProps, nextState, nextContext)
(skipped if forced update)
componentWillUpdate(nextProps, nextState, nextContext)
render()
componentDidUpdate(prevProps, prevState, prevContext)
Cannot use this.setState()
Cannot use this.setState()
Cannot use this.setState()
UNMOUNTING
UNMOUNTED
componentWillUnmount()
Cannot use this.setState()

2.4 Props & States

1.props: properties are passed to a component and can hold any data

```
class User extends Component {
  render(){
    const user = this.props.data;
    this.props.onReady('I am ready!');
    return(
      <View>
        <Text>
          score: {this.props.score}
          type: {this.props.type}
          Name: {user.name}
          Age: {user.age}
        </Text>
      </View>
    );
  }
}
//dufaultProps
User.propTypes = { score: React.PropTypes.number };
User.defaultProps = { score: 0 };

var user = {name: 'foo', age: 21};
class Main extends Component {
  handleReady(str){
    console.log(str);
  }
  render(){
    return(
      <View>
        <User type="Dev" data={user} onReady={this.handleReady}/>

      </View>
    );
  }
}
```

2.state: State differs from props in that it is internal to the component.

```
class Timer extends Component {
  constructor(props) {
    super(props);
    this.state = {count: 0};
  }

  componentDidMount() {
    let that = this;
    setInterval(function () {
      that.increase();
    }, 1000);
  }

  increase() {
    this.setState({count: this.state.count + 1});
  }

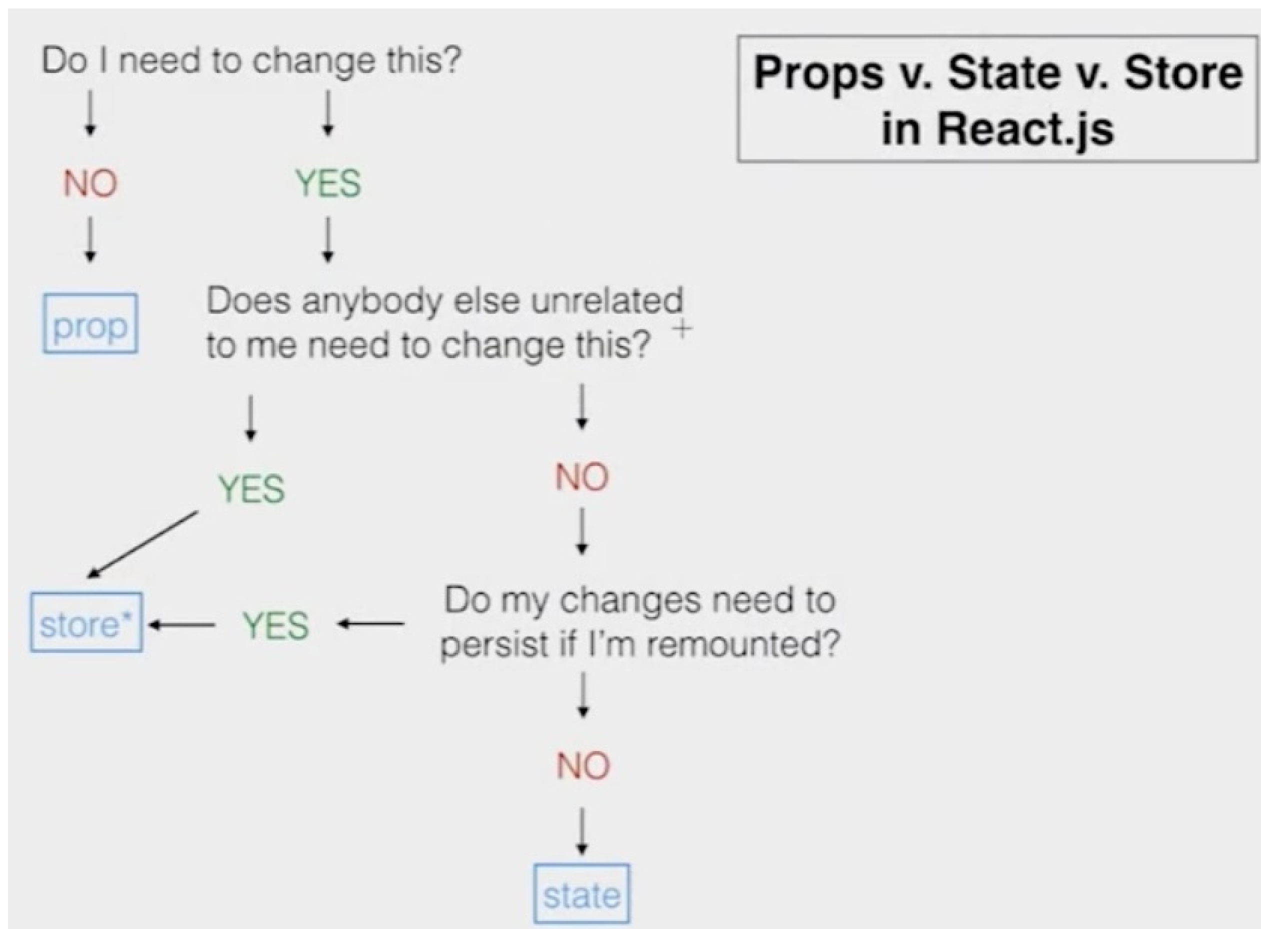
  render() {
    return (
      <View>
        <Text>count: {this.state.count}</Text>
      </View>
    );
  }
}

class Main extends Component {
  render(){
    return(
      <View>
        <Timer/>
      </View>
    );
  }
}
```

3.*props* VS *state*

- Use props to pass data and settings through the component tree.

- Never modify this.props inside of a component; consider props immutable.
- Use props to for event handlers to communicate with child components.
- Use state for storing simple view state like wether or not drop-down options are visible.
- Never modify this.state directly, use this.setstate instead.



4.Stateless Component

```
const Heading = ({title}) => <Text>{title}</Text>;

..
...
<Heading title="test title"/>
...
..
```

2.5 Events

1.Basic events

1.1. `<TouchableHighlight/>`

```
class Touch extends Component {
  handlePress(){
    console.log('press');
  }
  handleLongPress(){
    console.log('longPress');
  }
  render() {
    return (
      <TouchableHighlight
        onPress={this.handlePress}
        onLongPress={this.handleLongPress}>
        <View>
          <Text>Press me!</Text>
        </View>
      </TouchableHighlight>
    );
  }
}
```

1.2. `<TextInput/>`

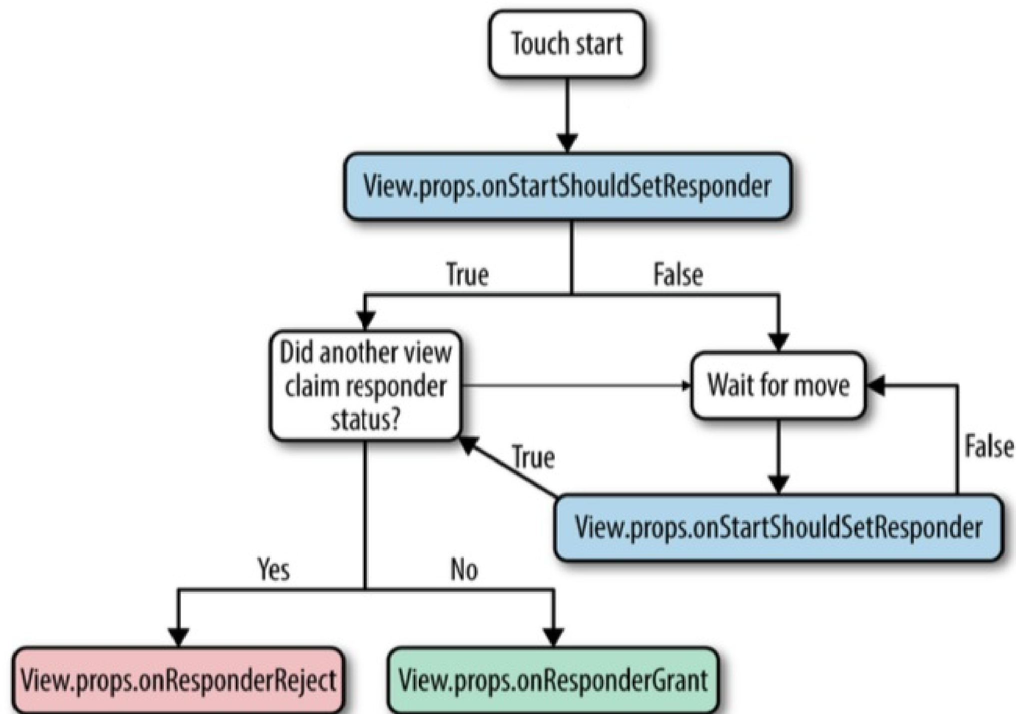
```
class Test extends Component {
  //...
  //handle events
  //...
  render() {
    return (
      <TextInput
        onBlur={...}
        onChange={...}
        onEndEditing={...}
        onSelectionChange={...}
        onSubmitEditing={...}
      </TextInput>
    );
  }
}
```

1.3.DeviceEventEmitter

```
//keyboardWillShow, keyboardDidShow, keyboardWillHide, keyboardD
idHide
//keyboardWillChangeFrame, keyboardDidChangeFrame
//add the listener
 var listener = DeviceEventEmitter.addListener('keyboardWillShow'
, (e) =>{
   console.log('Event is fired!');
 });
 //remove the listener
 listener.remove();
```

2.Gesture Responder System

2.1 Lifecycle



2.2 example

```
class Test extends Component {
  /* Capture handles */
  //the responder system bubbles up from the deepest component,
  //a parent View wants to prevent the child from becoming respo
nder on a touch start
  handleStartShouldSetResponderCapture(evt){
    return true;
  }
  //the responder system bubbles up from the deepest component,
  //a parent View wants to prevent the child from becoming respo
nder on a touch move
  handleMoveShouldSetResponderCapture(evt){
    return true;
  }

  /* Lifecycle handles */
  //Does this view want to become responder on the start of a to
uch?
  handleStartShouldSetResponder(evt){
    return true;
  }
```

```
  //Called for every touch move on the View when it is not the responder:
  //does this view want to "claim" touch responsiveness?
  handleMoveShouldSetResponder(evt){
    return true;
  }
  //The View is now responding for touch events.
  handleResponderGrant(evt){
    console.log('you are touching me');
  }
  //Something else is the responder right now and will not release it
  handleResponderReject(evt){
    console.log('please wait in line');
  }

  /* event handles */
  //touch move
  handleResponderMove(evt){
    console.log('touch move at:', 'X='+evt.pageX, 'Y='+evt.pageY);
  }
  //touch end/up
  handleResponderRelease(evt){
    console.log('touch end');
  }
  //Something else wants to become responder. Should this view release the responder?
  handleResponderTerminationRequest(evt){
    return true;
  }
  //touch cancel
  handleResponderTerminate(evt){
    console.log('touch canceled');
  }
  render() {
    return (
      <View
        onStartShouldSetResponderCapture={this.handleStartShouldSetResponderCapture}
```

```
        onMoveShouldSetResponderCapture={this.handleMoveShouldSetResponderCapture}
        onStartShouldSetResponder={this.handleStartShouldSetResponder}
        onMoveShouldSetResponder={this.handleMoveShouldSetResponder}
        onResponderGrant={this.handleResponderGrant}
        onResponderReject={this.handleResponderReject}
        onResponderMove={this.handleResponderMove}
        onResponderRelease={this.handleResponderRelease}
        onResponderTerminationRequest={this.handleResponderTerminationRequest}
        onResponderTerminate={this.handleResponderTerminate}>
          <Text>Press me!</Text>
      </View>
    );
  }
}
```

2.3 evt is a synthetic touch event with the following form nativeEvent:

- changedTouches - Array of all touch events that have changed since the last event
- identifier - The ID of the touch
- locationX - The X position of the touch, relative to the element
- locationY - The Y position of the touch, relative to the element
- pageX - The X position of the touch, relative to the root element
- pageY - The Y position of the touch, relative to the root element
- target - The node id of the element receiving the touch event
- timestamp - A time identifier for the touch, useful for velocity calculation
- touches - Array of all current touches on the screen

3.PanResponder

3.1

```
this._panResponder = PanResponder.create({
  // Ask to be the responder:
  onStartShouldSetPanResponder: (evt, gestureState) => true,
  onStartShouldSetPanResponderCapture: (evt, gestureState) => true,
  onMoveShouldSetPanResponder: (evt, gestureState) => true,
  onMoveShouldSetPanResponderCapture: (evt, gestureState) => true,
  //touch start
  onPanResponderGrant: (evt, gestureState) => {},
  //touch move
  onPanResponderMove: (evt, gestureState) => {},
  onPanResponderTerminationRequest: (evt, gestureState) => true,
  //touch end/up
  onPanResponderRelease: (evt, gestureState) => {},
  //touch cancel
  onPanResponderTerminate: (evt, gestureState) => {},
  onShouldBlockNativeResponder: (evt, gestureState) => true,
});
```

3.2 A gestureState object has the following:

- stateID - ID of the gestureState- persisted as long as there at least one touch on screen
- moveX - the latest screen coordinates of the recently-moved touch
- moveY - the latest screen coordinates of the recently-moved touch
- x0 - the screen coordinates of the responder grant
- y0 - the screen coordinates of the responder grant
- dx - accumulated distance of the gesture since the touch started
- dy - accumulated distance of the gesture since the touch started
- vx - current velocity of the gesture
- vy - current velocity of the gesture
- numberActiveTouches - Number of touches currently on screen

3.3 PanResponder example in UIExplorer

2.6 Resources

- react.parts
- js.coach
- props vs state
- Thinking in React
- JSX in Depth
- DEMO scripts for this chapter

3 Styles

1.Declare Style

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: 'blue',
  },
  text: {
    fontSize: 14,
    color: 'red'
  }
});
```

2.Using Styles

```
class Main extends Component {
  render() {
    return (
      <View style={styles.container}>
        <Text style={styles.text}>I am red.</Text>
      </View>
    );
  }
}
```

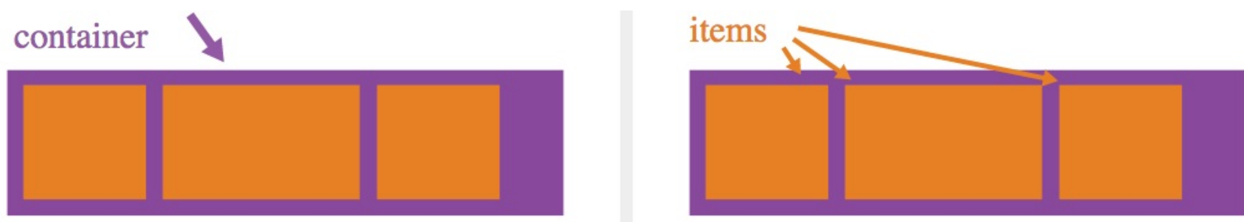
3.Properties

- View Properties
- Image Properties
- Text Properties
- Flex Properties
- Transform Properties

3.1 Flexbox

1.Flexbox layout

> The main idea behind the flex layout is to give the container the ability to alter its items' width/height (and order) to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes). A flex container expands items to fill available free space, or shrinks them to prevent overflow.

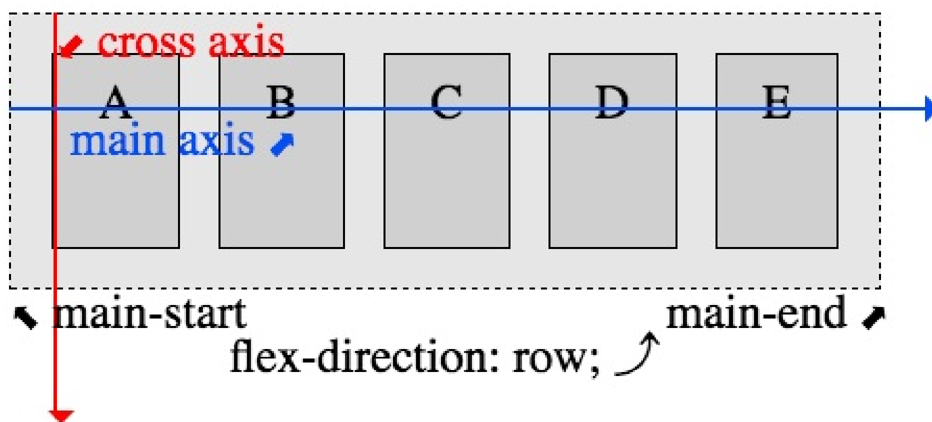


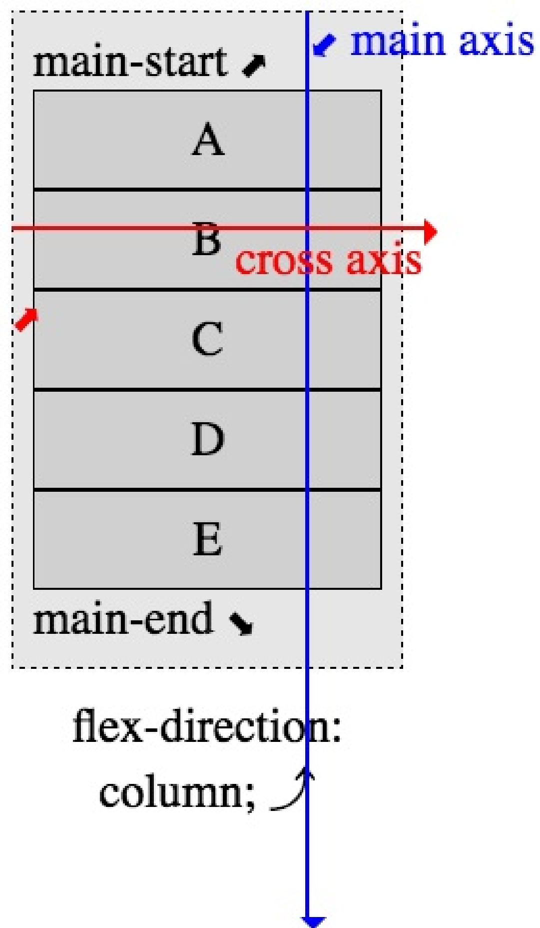
2.flex:1



```
const styles = StyleSheet.create({
  container: {
    flex: 1
  },
  header: {
    height: 200,
    backgroundColor: 'red'
  },
  main: {
    flex: 1,
    backgroundColor: 'blue'
  },
  footer: {
    height: 200,
    backgroundColor: 'green'
  },
  text: {
    color: '#ffffff',
    fontSize: 80
  }
});
```

3.flexDirection:'row'|'column'





4.justifyContent:'flex-start'|'flex-end'|'center'|'space-between'|'space-around'

justify-content

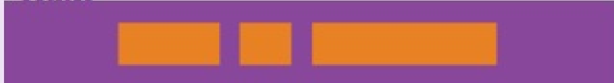
flex-start



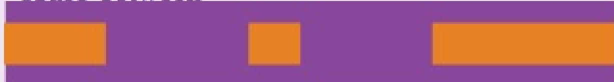
flex-end



center



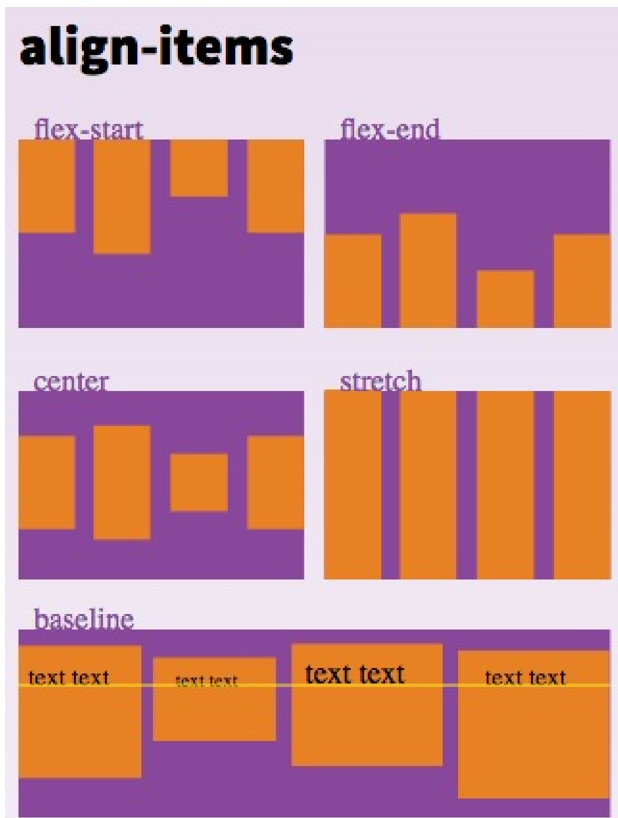
space-between



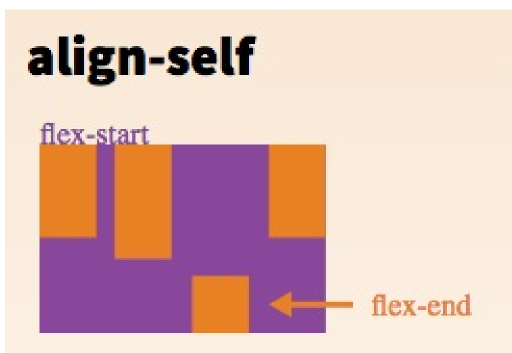
space-around



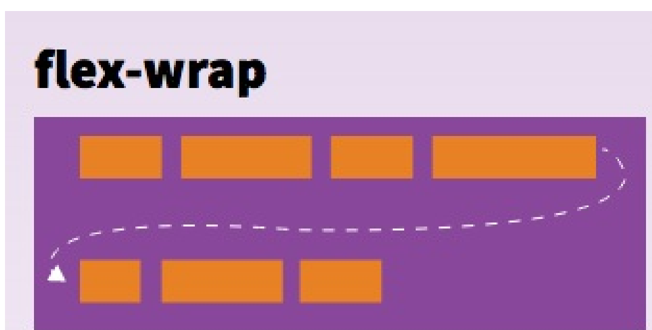
5.alignItems:'flex-start'|'flex-end'|'center'|'stretch'



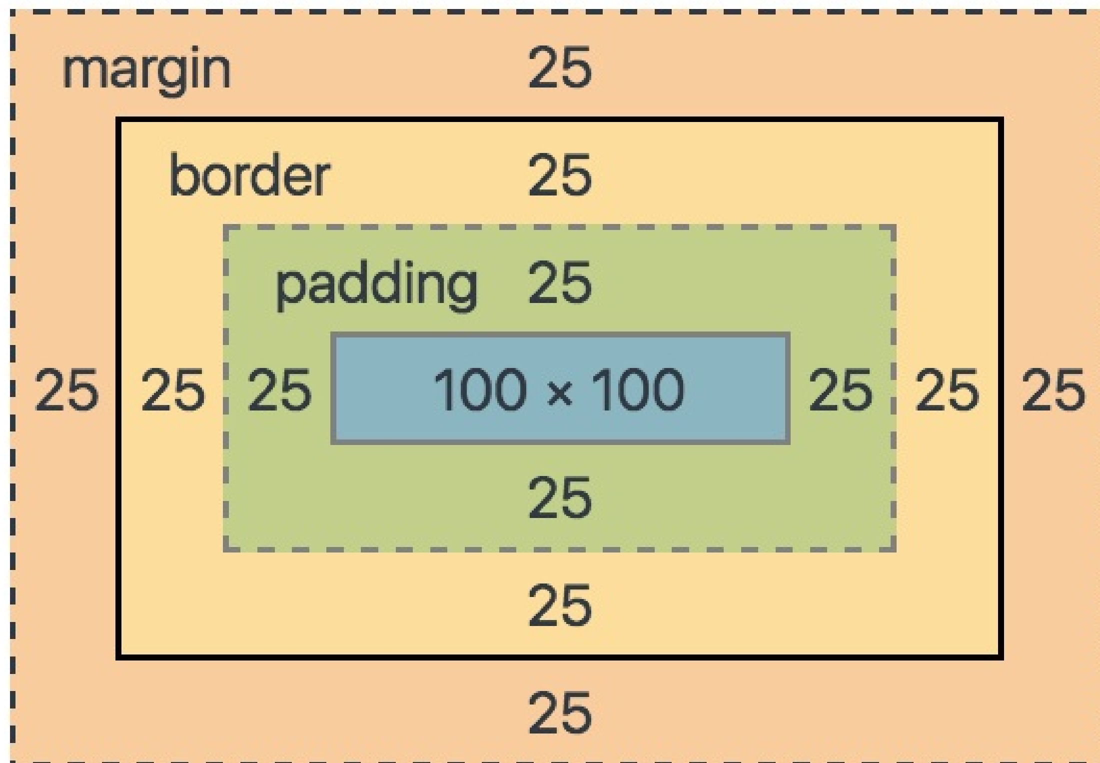
6.alignSelf:'auto'|'flex-start'|'flex-end'|'center'|'stretch'



7.flexWrap:'wrap'|'nowrap'



8.Box model



width =
borderLeftWidth(25)+paddingLeft(25)+100+borderRightWidth(25)+paddingRight(25)=200

height =
borderTopWidth(25)+paddingTop(25)+100+borderBottomWidth(25)+paddingBottom(25)=200

```
class Main extends Component {
  render() {
    return (
      <View style={styles.container}>
        <View style={styles.header}>
          <Text style={styles.text}>200X100</Text>
        </View>
        <View style={styles.main}>
          <View  style={styles.mainContent}>
            <Text style={styles.text}>100X100</Text>
          </View>
        </View>
        <View style={styles.footer}>
          <Text style={styles.text}>200X100</Text>
```

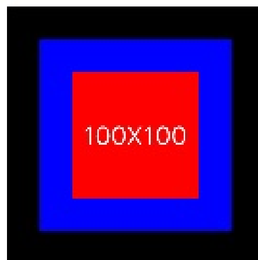
```
        </View>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center'
  },
  header: {
    height: 100,
    width: 200,
    backgroundColor: 'red'
  },
  main: {
    height: 200,
    width: 200,
    padding: 25,
    borderWidth: 25,
    borderColor: 'black',
    margin: 25,
    backgroundColor: 'blue'
  },
  mainContent: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: 'red'
  },
  footer: {
    height: 100,
    width: 200,
    backgroundColor: 'green'
  },
  text: {
    color: '#ffffff',
    fontSize: 20
```

```
    }
});
```


iPhone 6 - iPhone 6 / iOS...
Carrier
5:45 PM


200X100

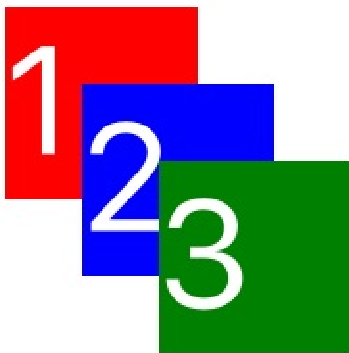

100X100


200X100

3.2 Absolute & Relative

1.absolute






```
class Position extends Component {
  render() {
    return (
      <View style={styles.container}>
        <View style={styles.box1}>
          <Text style={styles.text}>1</Text>
        </View>
        <View style={styles.box2}>
          <Text style={styles.text}>2</Text>
        </View>
        <View style={styles.box3}>
          <Text style={styles.text}>3</Text>
        </View>
      </View>
    );
  }
}
const styles = StyleSheet.create({
  container: {
    flex: 1
```

```
  },
  box1: {
    position: 'absolute',
    top: 40,
    left: 40,
    width: 100,
    height: 100,
    backgroundColor: 'red'
  },
  box2: {
    position: 'absolute',
    top: 80,
    left: 80,
    width: 100,
    height: 100,
    backgroundColor: 'blue'
  },
  box3: {
    position: 'absolute',
    top: 120,
    left: 120,
    width: 100,
    height: 100,
    backgroundColor: 'green'
  },
  text: {
    color: '#ffffff',
    fontSize: 80
  }
});
```

2.zIndex, v0.29 or transform




```
box2: {
  position: 'absolute',
  top: 80,
  left: 80,
  width: 100,
  height: 100,
  backgroundColor: 'blue',
  transform: [{'translate': [0,0, 1]}]
},
```

3.relative(default)

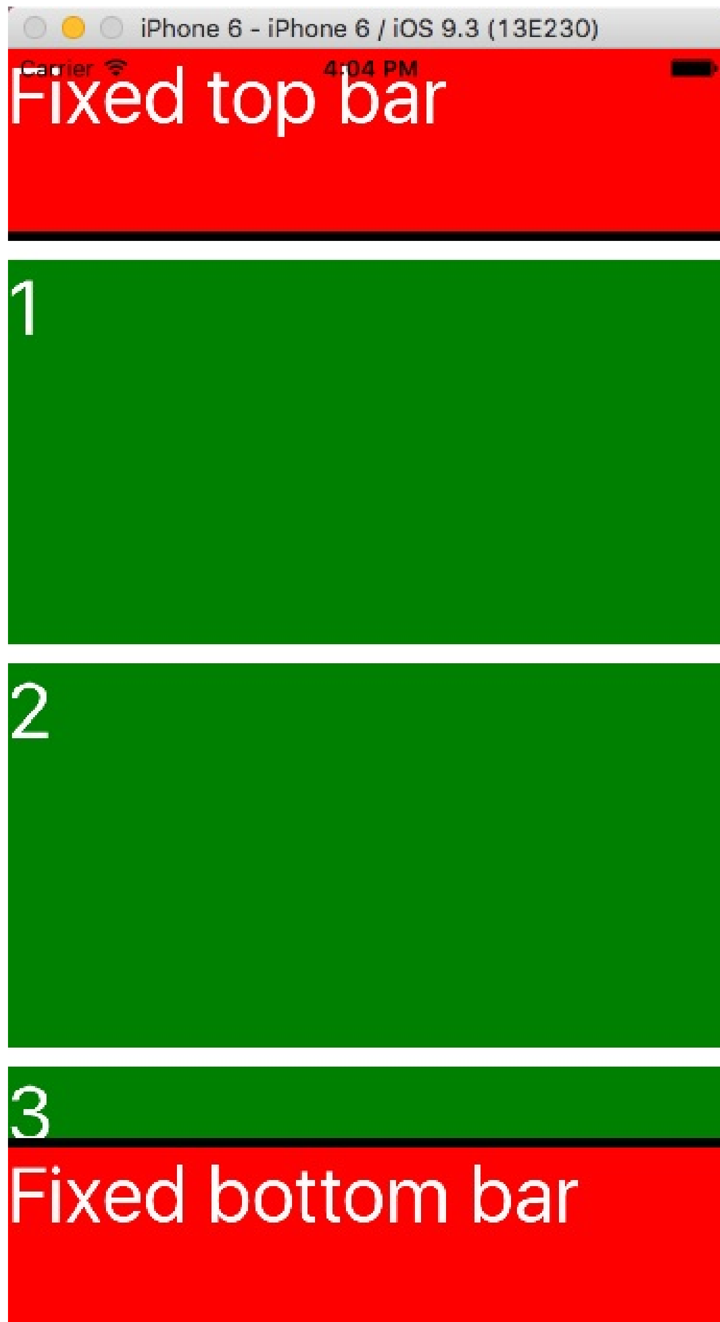



```
class Relative extends Component {
  render() {
    return (
```

```
        <View style={styles.container}>
          <View style={styles.box1}>
            <Text style={styles.text}>1</Text>
            <View style={styles.ball}/>
          </View>
          <View style={styles.box2}>
            <Text style={styles.text}>2</Text>
          </View>
        </View>
      );
    }
  }
const styles = StyleSheet.create({
  container: {
    flex: 1
  },
  box1: {
    position: 'relative',
    top: 40,
    left: 40,
    width: 100,
    height: 100,
    backgroundColor: 'red'
  },
  box2: {
    position: 'absolute',
    top: 100,
    left: 100,
    width: 100,
    height: 100,
    backgroundColor: 'blue'
  },
  ball: {
    position: 'absolute',
    top: 40,
    left: 40,
    width: 40,
    height: 40,
    borderRadius: 20,
    backgroundColor: 'yellow'
```

```
  },
  text: {
    color: '#ffffff',
    fontSize: 80
  }
});
```

4.fixed



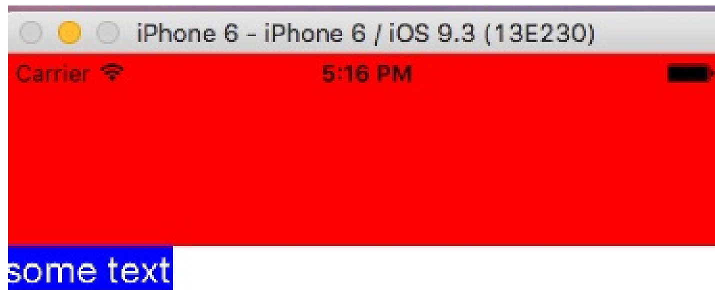

```
class Fixed extends Component {
```

```
  render() {
    return (
      <View style={styles.container}>
        <View style={styles.tbar}>
          <Text style={styles.text}>Fixed top bar</Text>
        </View>
        <ScrollView style={styles.main}>
          <View style={styles.item}><Text style={styles.text}>1</
Text></View>
          <View style={styles.item}><Text style={styles.text}>2</
Text></View>
          <View style={styles.item}><Text style={styles.text}>3</
Text></View>
        </ScrollView>
        <View style={styles.bbar}>
          <Text style={styles.text}>Fixed bottom bar</Text>
        </View>
      </View>
    );
  }
}
const styles = StyleSheet.create({
  container: {
    flex: 1
  },
  tbar: {
    width: 375,
    height: 100,
    borderBottomWidth: 5,
    borderColor: 'black',
    backgroundColor: 'red'
  },
  main: {
    flex: 1
  },
  item: {
    height: 200,
    width: 375,
    marginTop: 10,
    backgroundColor: 'green'
```

```
    },
    bbar: {
      width: 375,
      height: 100,
      borderTopWidth: 5,
      borderColor: 'black',
      backgroundColor: 'red'
    },
    text: {
      color: '#ffffff',
      fontSize: 40
    }
  });
```

3.3 Size & Dimensions & onLayout

1.window size



```
let winSize = Dimensions.get('window');
console.log(winSize);
class Size extends Component {
  render() {
    return (
      <View style={styles.container}>
        <View style={styles.block}/>
        <Text style={styles.text}>some text</Text>
      </View>
    );
  }
}
const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'flex-start'
  },
  block: {
    height: 100,
    width: winSize.width,
    backgroundColor: 'red'
  },
  text: {
    color: '#ffffff',
    fontSize: 40/winSize.scale,
    backgroundColor: 'blue'
  }
});
```

2.onLayout

```
class Size extends Component {
  handleTextLayout(evt){
    console.log(evt.nativeEvent.layout);
  }
  render() {
    return (
      <View style={styles.container}>
        <View style={styles.block}/>
        <Text style={styles.text}
          onLayout={this.handleTextLayout}>some text</Text>
      </View>
    );
  }
}
```

3.4 Inheritance

1.pass styles as props

```
class InheritanceStyle extends Component {
  render() {
    return (
      <View style={this.props.parentColor}>
      </View>
    );
  }
}

class Main extends Component {
  handleReady(str){
    console.log(str);
  }
  render() {
    return (
      <View style={styles.container}>
        <InheritanceStyle parentColor={styles.blue}/>
      </View>
    );
  }
}
const styles = StyleSheet.create({
  container: {
    flex: 1
  },
  blue: {
    flex: 1,
    backgroundColor: 'blue'
  }
});
```

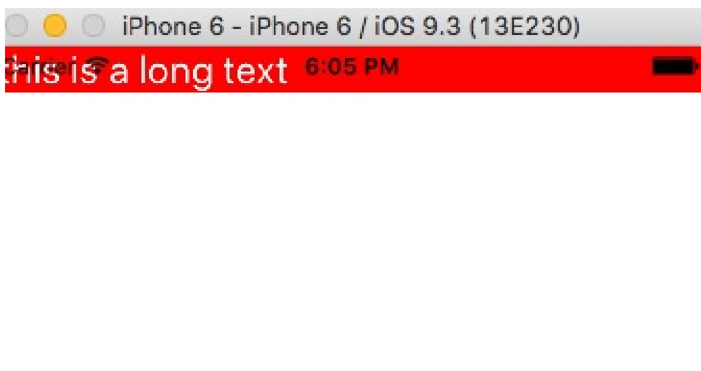
2.concatenation styles

BaseStyles.js

```
import { StyleSheet,Dimensions } from 'react-native';
let winSize = Dimensions.get('window');
const BaseStyles = StyleSheet.create({
  text: {
    fontSize: 40/winSize.scale
  }
});
export default BaseStyles;
```

```
import BaseStyles from './BaseStyles';

class InheritanceStyle extends Component {
  render() {
    return (
      <View style={this.props.parentColor}>
        <Text style={[BaseStyles.text, styles.text]}> this is a
long text </Text>
      </View>
    );
  }
}
const styles = StyleSheet.create({
  text:{
    color: '#ffffff'
  }
});
```

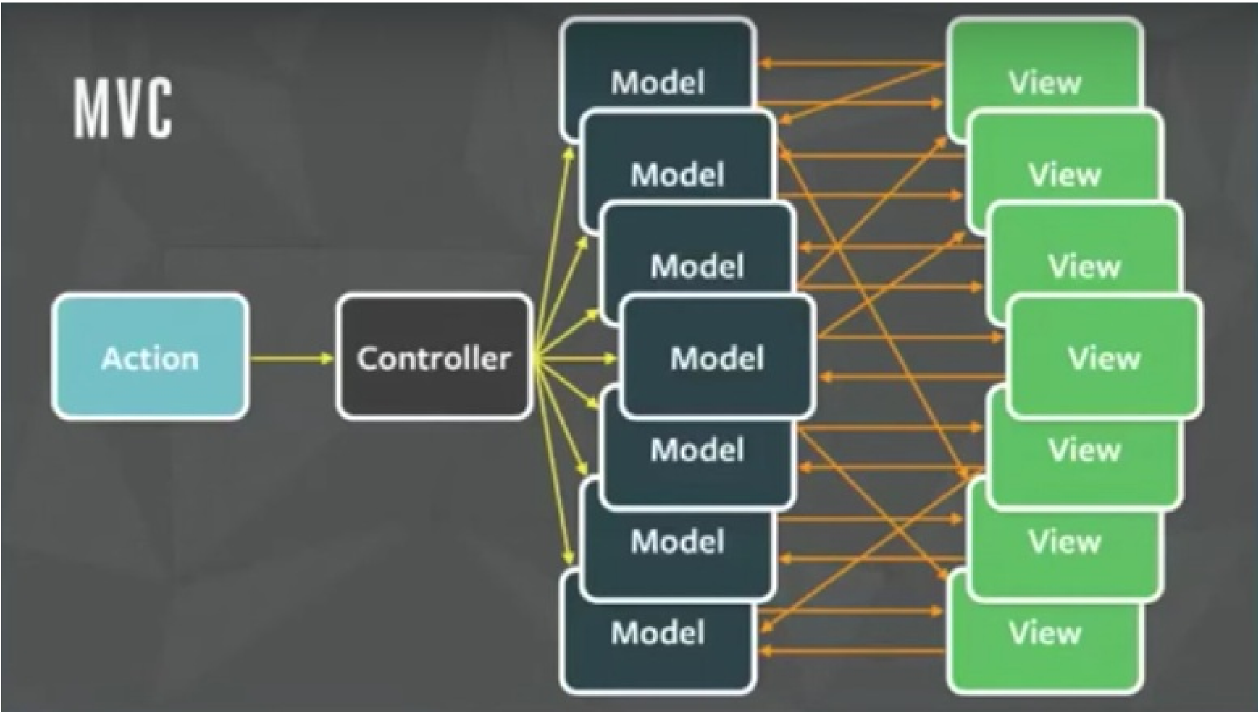


3.5 Resources

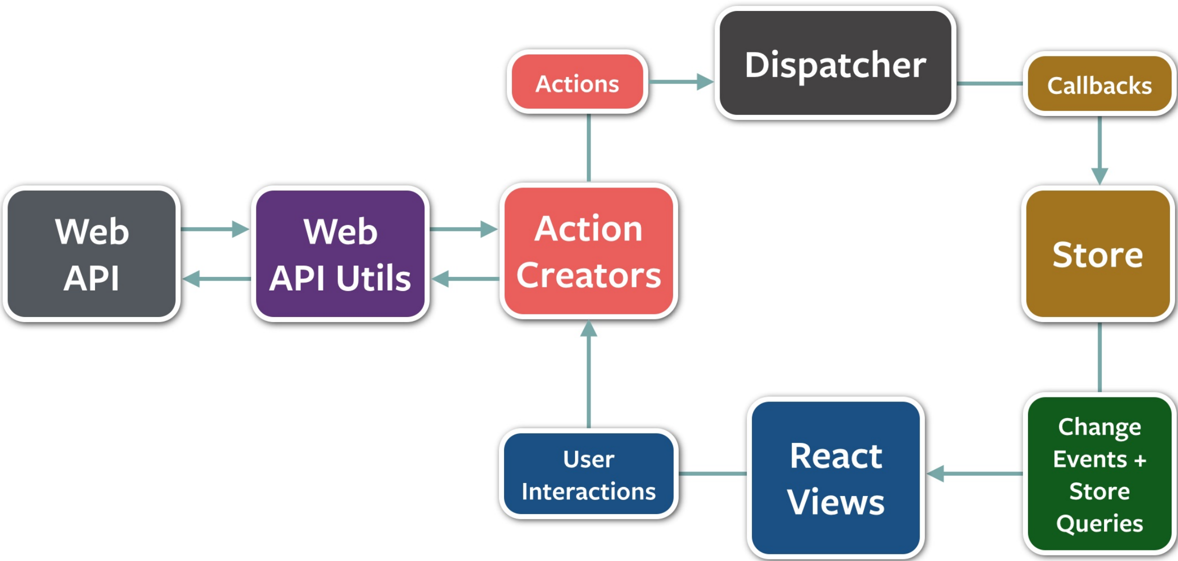
- A Complete Guide to Flexbox
- A Visual Guide to CSS3 Flexbox Properties
- Understanding Flex Direction
- DEMO scripts for this chapter

4 Architecture

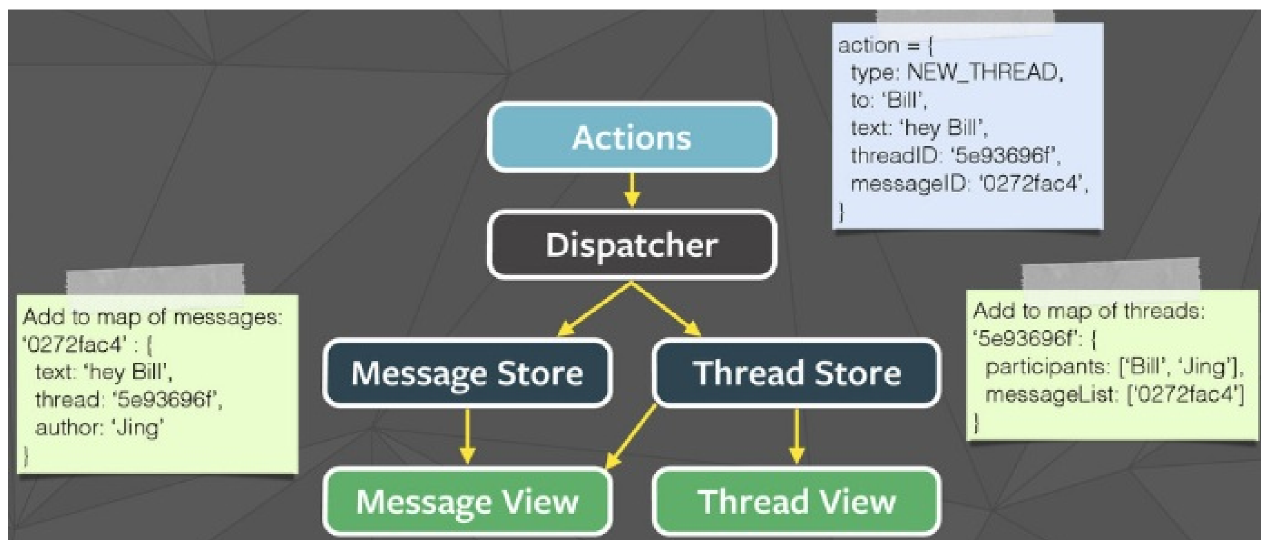
1.MVC problems



2.Flux



3.Data flow



[Flux TodoMVC Example]

4.1 Redux

1.Actions & Action Creators

```
//action type
const ADD_TODO = 'ADD_TODO';

//action creator, semantic methods that create actions
//collected together in a module to become an API
function addTodoAction(title, hour) {
  //action, an object with a type property and new data, like ev
ents
  return {type: ADD_TODO, title, hour}
}
```

2.Reducers

```
//a function that accepts an accumulation and a value and return
s a new accumulation.
function todoReducers(state = [], action) {
  switch (action.type) {
    case ADD_TODO:
      //always return a new state, never mutate old state
      return [
        {
          id: Utils.GUID(),
          title: action.title,
          endTime: getEndTime(action.hour),
          completed: false
        },
        ...state
      ]
    default:
      //return default state
      return state
  }
}
```

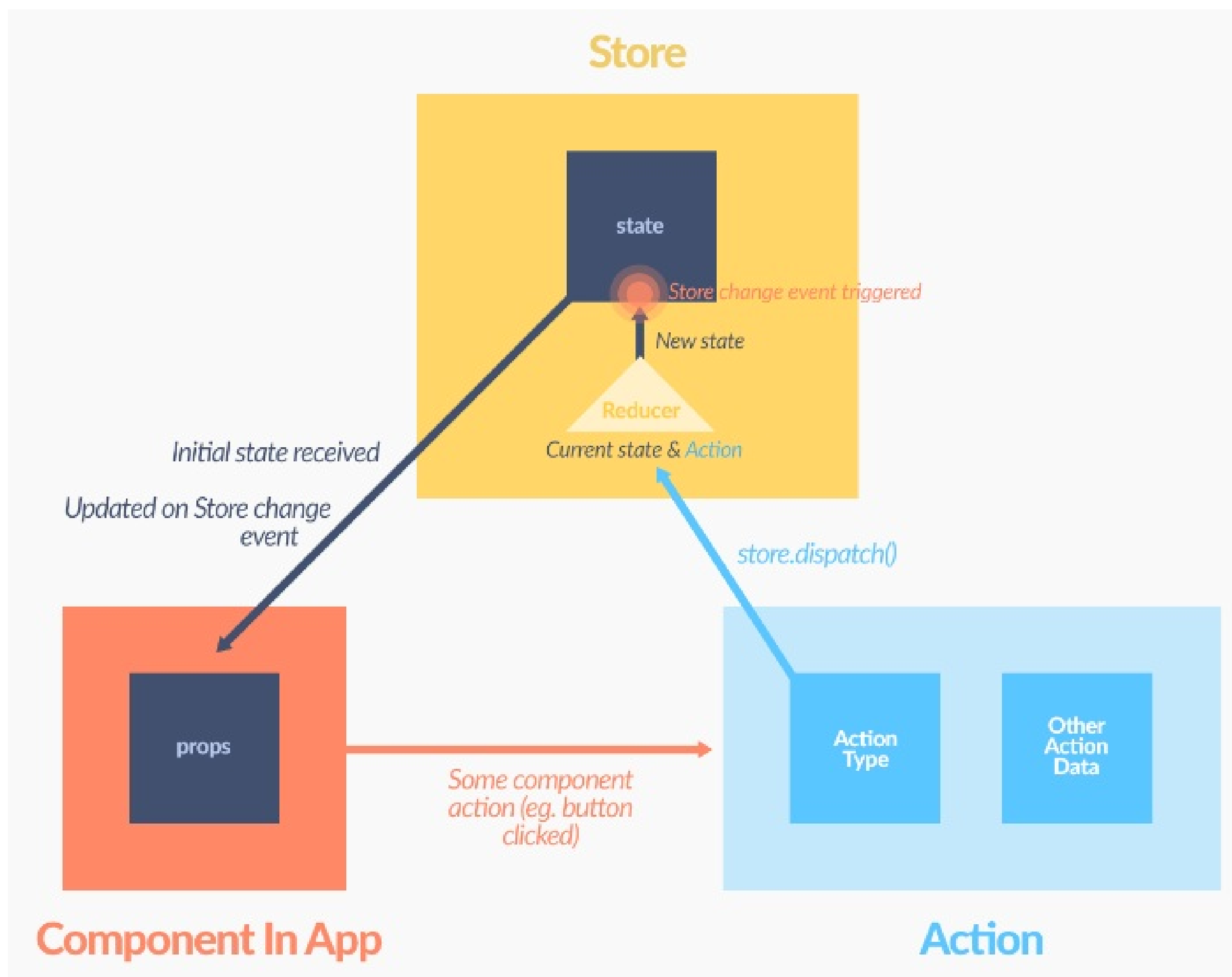
3.Store

```
import { createStore } from 'redux';
//1. define store
let store = createStore(todoReducers);

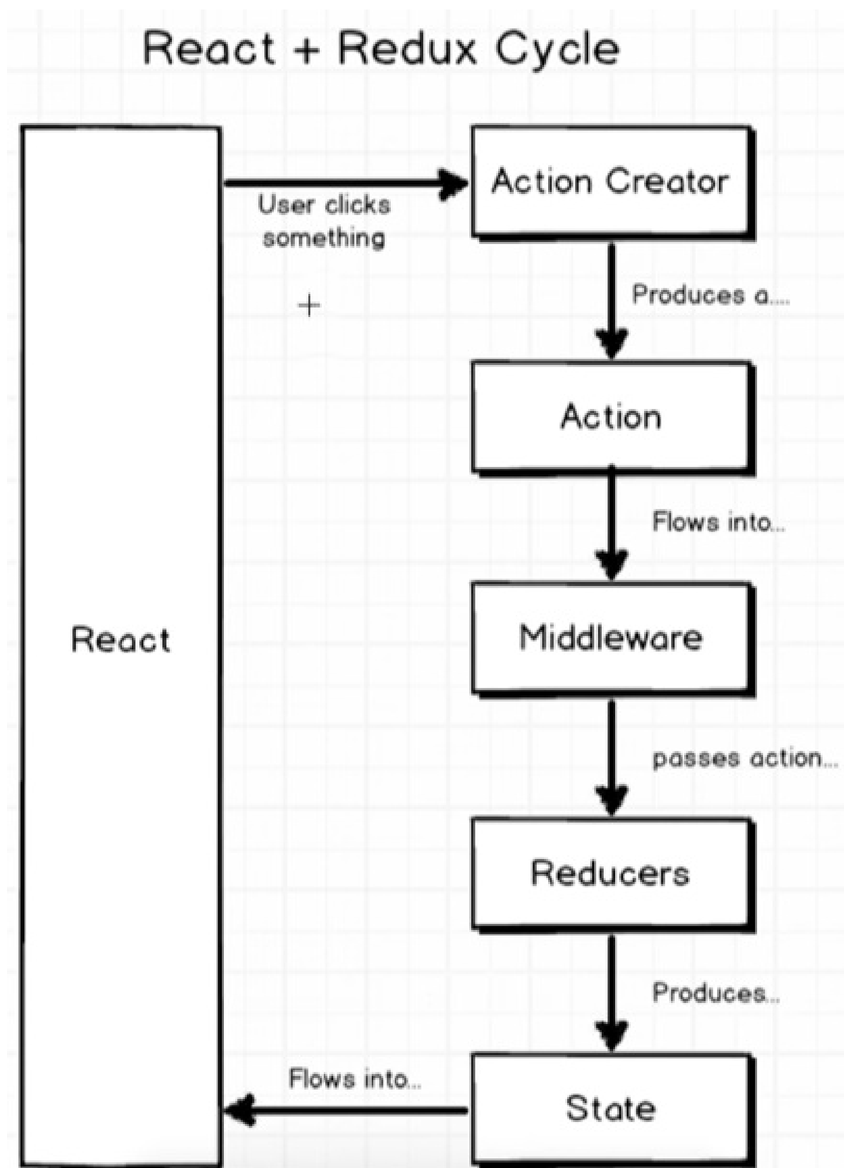
class App extends Component {
  constructor(props){
    super(props);
    this.state = {todos: []};
  }
  componentDidMount(){
    //2. subscribe store
    this.unsubscribeStore = store.subscribe(() =>{
      //3. getState
      this.setState({todos: store.getState()});
    });
  }
```

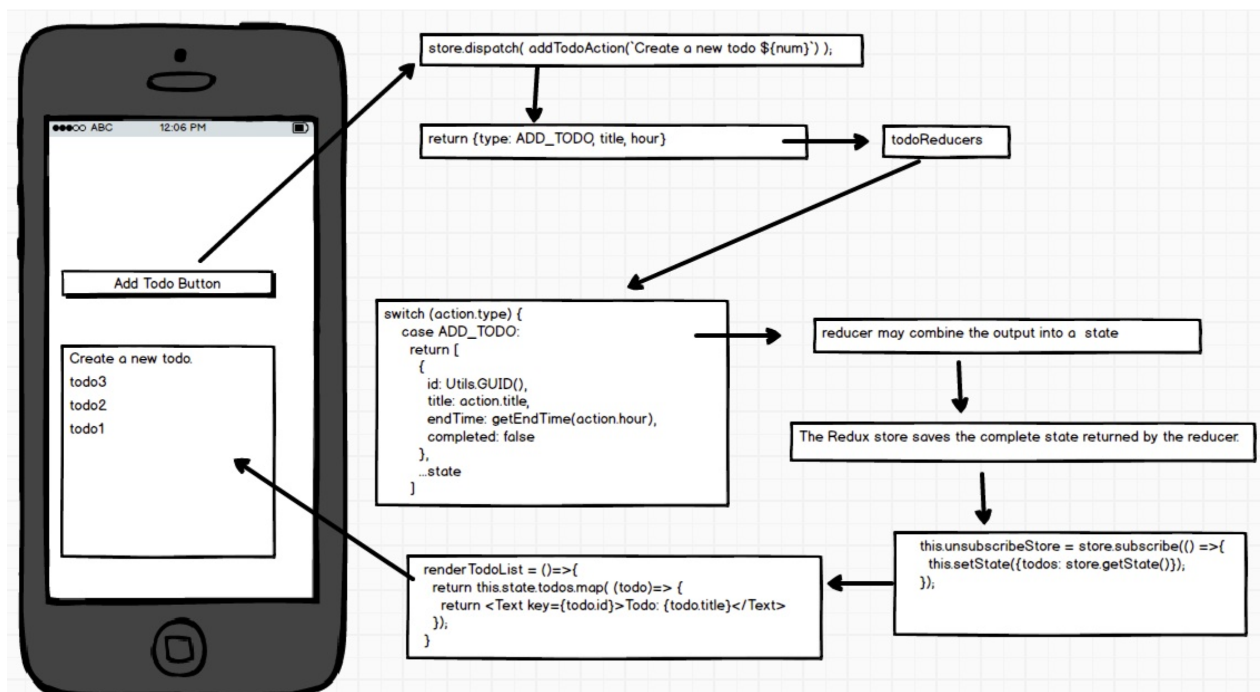
```
    componentWillUnmount(){
      //5. unsubscribe store
      this.unsubscribeStore();
    }
    renderTodoList = ()=>{
      //reder todo list
      return this.state.todos.map( (todo)=> {
        return <Text key={todo.id}>Todo: {todo.title}</Text>
      });
    }
    handleAddTodo = ()=>{
      //4. dispatching actions
      store.dispatch( addTodoAction('Create a new todo', 8) );
    }
    render() {
      return (
        <View>
          <TouchableHighlight onPress={this.handleAddTodo}>
            <Text>Add Todo</Text>
          </TouchableHighlight>
          <ScrollView>{this.renderTodoList()}</ScrollView>
        </View>
      );
    }
  }
```

4.Data flow



Store
state
Store change event triggered
New state
Reducer
Current state & Action
Initial state received
Updated on Store change event
store.dispatch()
props
Some component action (eg. button clicked)
Action Type
Other Action Data
Component In App
Action


React + Redux Cycle
React
User clicks something
Action Creator
Produces a....
Action
Flows into...
Middleware
passes action...
Reducers
Produces...
State
Flows into...


ABC
12:06 PM
Add Todo Button
Create a new todo.
todo3
todo2
todo1
store.dispatch(addTodoAction(`Create a new todo ${num}`));
return {type: ADD_TODO, title, hour}
todoReducers
switch (action.type) {
case ADD_TODO:
return [
{
id: Utils.GUID(),
title: action.title,
endTime: getEndTime(action.hour),
completed: false
},
...state
]
reducer may combine the output into a state
The Redux store saves the complete state returned by the reducer.
this.unsubscribeStore = store.subscribe(() =>{
this.setState({todos: store.getState()});
});
renderTodoList = ()=>{
return this.state.todos.map((todo)=> {
return <Text key={todo.id}>Todo: {todo.title}</Text>
});
}

4.2 react-redux

1.Actions

```
import * as  navigationActions from './navigation';
import * as  todosActions from './todos';

export default {...navigationActions, ...todosActions};
```

2.combineReducers()

```
import { combineReducers } from 'redux';
import navigation from './navigation';
import todos from './todos';

const rootReducer = combineReducers({
  navigation, todos
});

export default rootReducer;
```

3.Application state by configureStore()

```
import { createStore } from 'redux';
import reducers from '../reducers';

export default function configureStore() {
  const store = createStore(reducers);
  return store;
}
```

4.mapStateToProps & mapDispatchToProps & bindActionCreators

```
import { bindActionCreators } from 'redux';
import { connect } from 'react-redux';
```

```
class App extends Component {
  renderTodoList = ()=>{
    //reder todo list
    return this.props.todos.map( (todo)=> {
      return <Text key={todo.id}>Todo: {todo.title}</Text>
    });
  }
  handleAddTodo = ()=>{
    this.props.actions.addTodoAction('Create a new todo', 8);
  }
  render() {
    return (
      <View>
        <TouchableHighlight onPress={this.handleAddTodo}>
          <Text>Add Todo</Text>
        </TouchableHighlight>
        <ScrollView>{this.renderTodoList()}</ScrollView>
      </View>
    );
  }
}

function mapStateToProps(state) {
  return {
    todos: state.todos
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(Actions, dispatch)
  }
}

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(App);
```

5.Passing the Store with `<Provider/>`

```
import React, { Component } from 'react';
import { Provider } from 'react-redux';

import App from './containers/App';
import configureStore from './store/configureStore';

class Root extends Component {
  render() {
    return (
      <Provider store={configureStore()}>
        <App />
      </Provider>
    );
  }
}

export default Root;
```

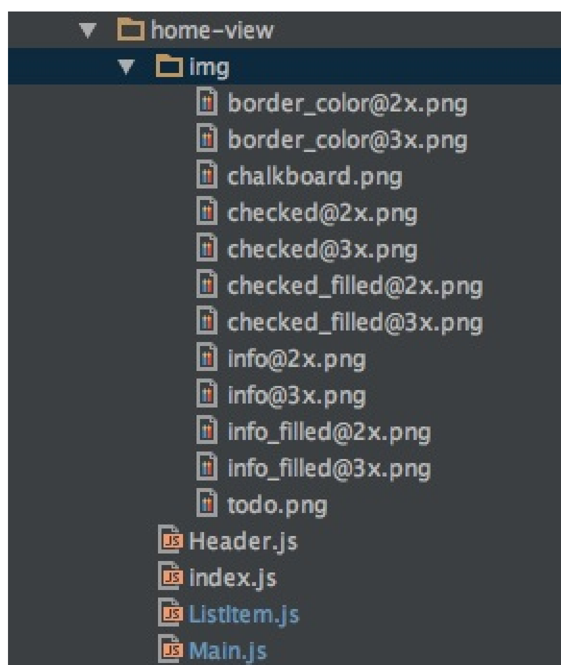
4.3 Containers & Components

1.Presentational and Container Components

	Presentational Components	Container Components
Purpose	How things look (markup, styles)	How things work (data fetching, state updates)
Aware of Redux	No	Yes
To read data	Read data from props	Subscribe to Redux state
To change data	Invoke callbacks from props	Dispatch Redux actions
Are written	By hand	Usually generated by React Redux

2. `components/home-view` & `containers/HomeView`

2.1 home-view components



2.2 HomeView container

```
import {
  Header,
  Main,
} from '../components/home-view';
import Actions from '../actions';

class HomeView extends Component {
  render() {
    return (
      <View>
        <Header {...this.props}/>
        <Main {...this.props} isVisible={this.state.isVisible}/>
      </View>
    );
  }
}

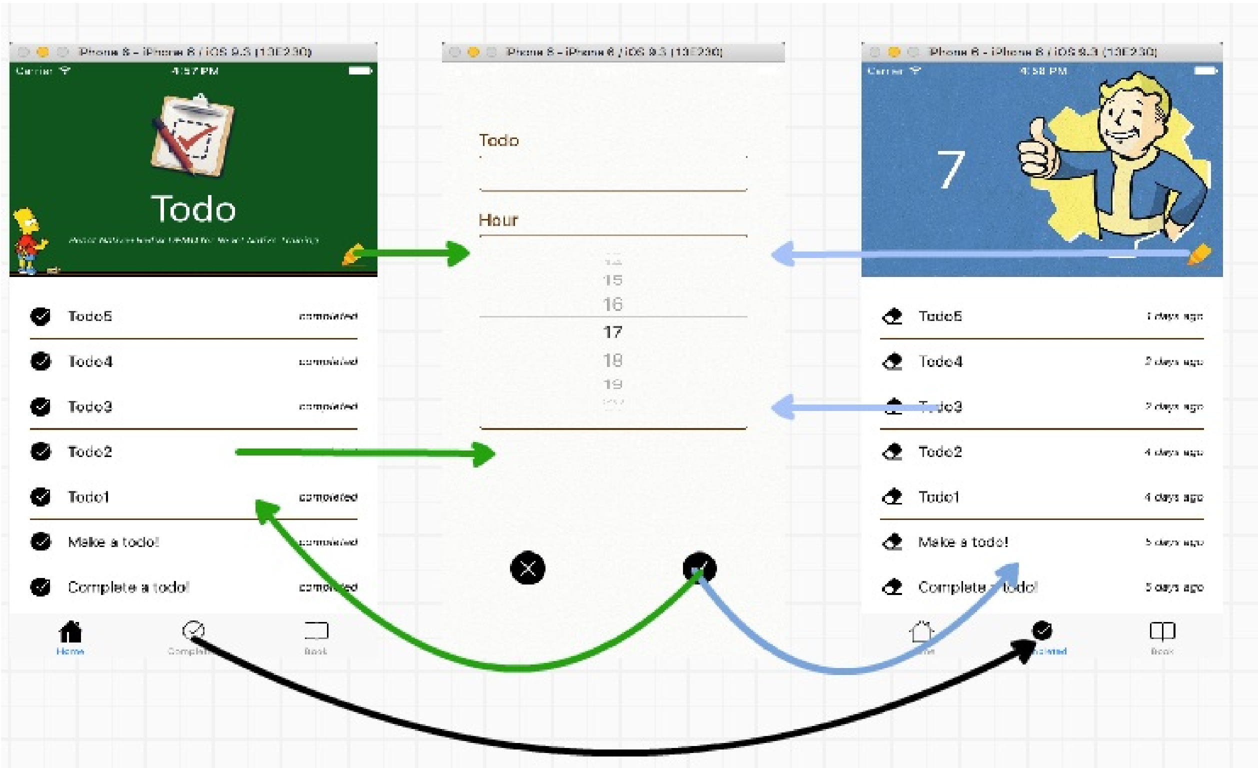
function mapStateToProps(state) {
  return {
    todos: state.todos
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(Actions, dispatch)
  }
}

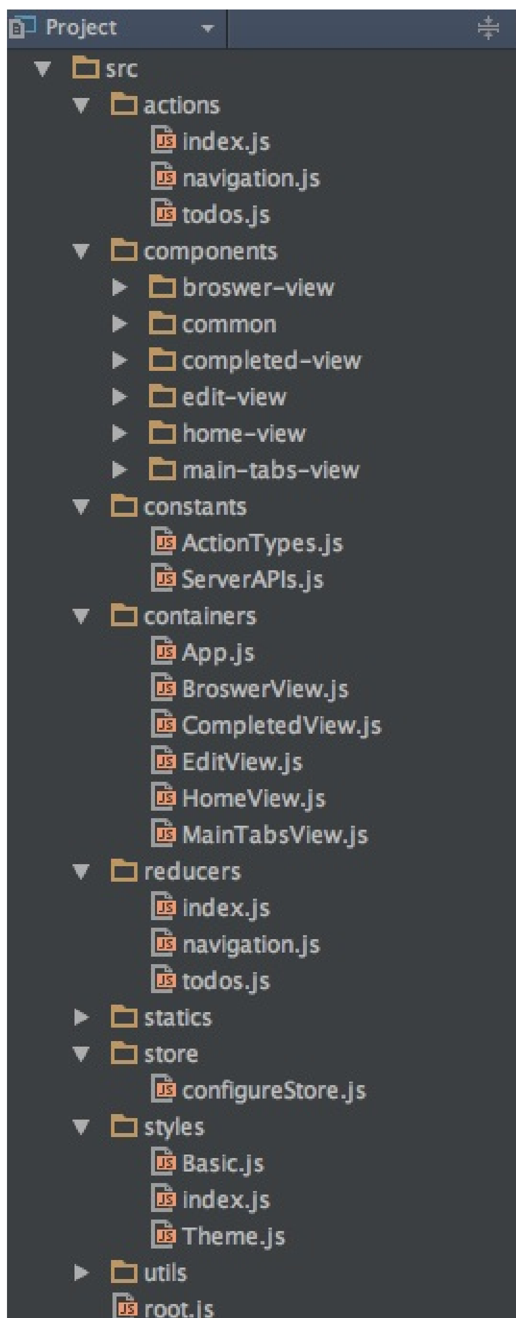
export default connect(
  mapStateToProps,
  mapDispatchToProps
)(HomeView);
```

4.4 Todo React Native App

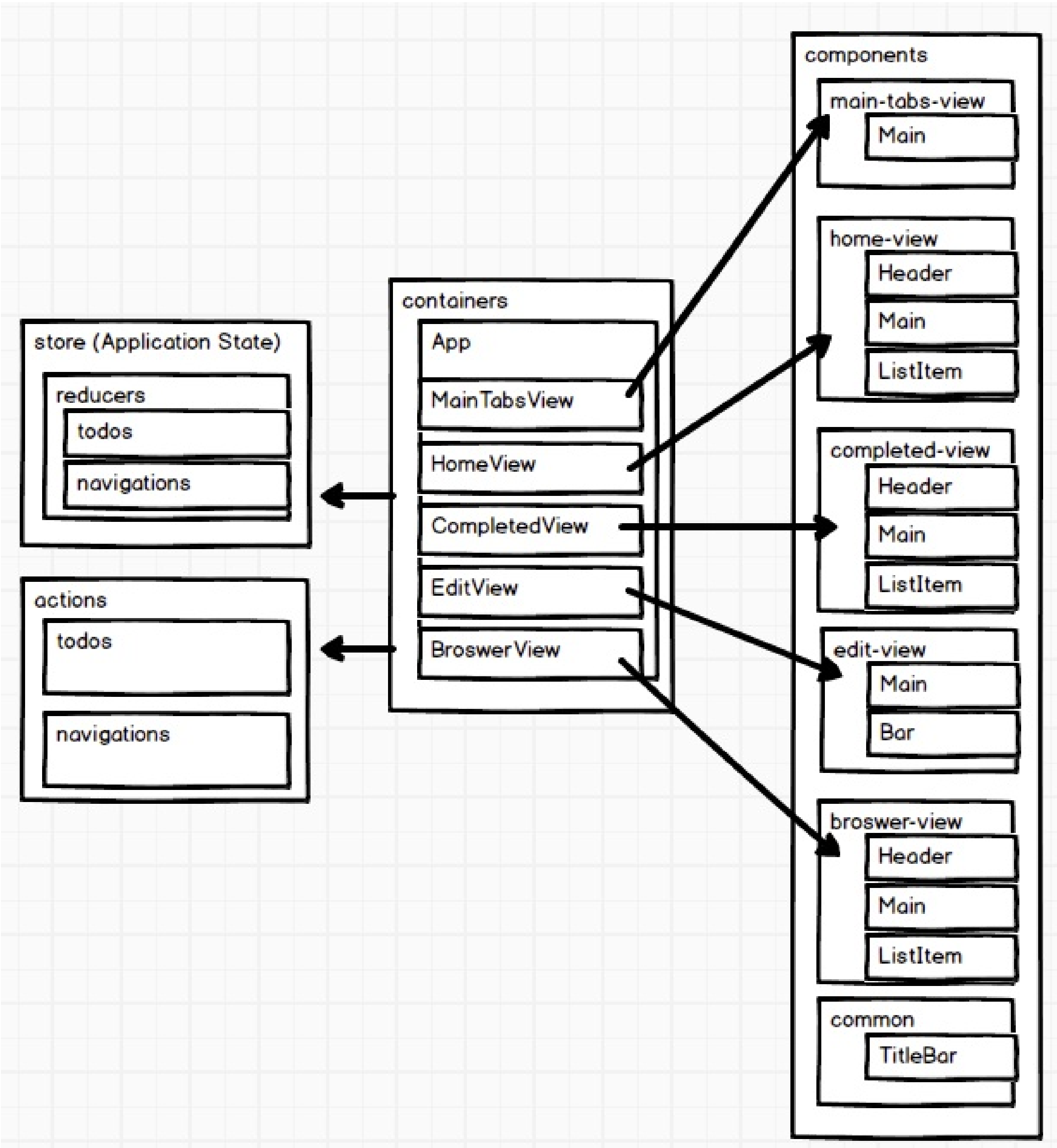
1.Overview



2.Structure


Project
src
actions
index.js
navigation.js
todos.js
components
broswer-view
common
completed-view
edit-view
home-view
main-tabs-view
constants
ActionTypes.js
ServerAPIs.js
containers
App.js
BroswerView.js
CompletedView.js
EditView.js
HomeView.js
MainTabsView.js
reducers
index.js
navigation.js
todos.js
statics
store
configureStore.js
styles
Basic.js
index.js
Theme.js
utils
root.js

3.Containers & Components



4.5 Naming convention

1.Containers & Components

1.1. Container file:

```
src/containers/ModuleNameView.js
```

Component files:

```
src/components/module-name-view
 - index.js
 - Main.js
 - Header.js
 - ...
 - img
   - icon@2x.png
   - icon@3x.png
```

1.2. Event name:

```
handleEventName = ()=>{//todo}
...
<MyComponent onEventName={this.handleEventName}/>
```

1.3. Render methods:

```
  renderMethodName = () => {
   //todo
  }
  render() {
    return (
      <View>
        {this.renderMethodName()}
      </View>
    );
  }
`
```

1.4. mapStateToProps & mapDispatchToProps

```
function mapStateToProps(state) {
  return {
    todos: state.todos
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(Actions, dispatch)
  }
}
```

2.actions `src/actions`

```
index.js
todos.js
navigation.js
```

2.1 `src/constants/ActionTypes.js`

```
export const SWITCH_MAIN_TAB = 'SWITCH_MAIN_TAB';
```

2.2```src/actions/todos.js````

```
import * as types from '../constants/ActionTypes'

export function addTodo(title, hour) {
  return {type: types.ADD_TODO, title, hour}
}
```

3.reducers `src/reducers`

```
index.js
todos.js
navigation.js
```

3.1. `src/reducers/todos.js`

```
import { ADD_TODO, DELETE_TODO, EDIT_TODO, COMPLETE_TODO } from
'../constants/ActionTypes'
const initialState = []

export default function todos(state = initialState, action) {
  switch (action.type) {
    case ADD_TODO:
      //todo
    default:
      return state
  }
}
```

4.styles `src/styles`

```
index.js
Basic.js
Theme.js
```

4.1 `src/styles/Basic.js`

```
import { StyleSheet, Dimensions } from 'react-native';
let winSize = Dimensions.get('window');
const Basic = StyleSheet.create({
  text: {
    fontSize: 32/winSize.scale
  }
});
export default Basic;
```

4.2 `src/styles/Theme.js`

```
//colors
const color = {
  green: '#00551e',
  brown: '#693504',
  red: '#db2828'
}

//other
const active = {
  opacity: 0.6
}

export default {color, active}
```

4.3 `import {Theme, BasicStyle} from '../../styles';`

4.7 Resources

- Flux
- Facebook: MVC Does Not Scale, Use Flux Instead
- Redux
- fluxchat by Bill Fisher
- Introduce Flux & react in practices (KKBOX)
- react-flux-fluent-2015 by Bill Fisher
- Flux TodoMVC Example
- Todo React Native App

5 Data

1.Networking

- Fetch
- XMLHttpRequest API
- WebSocket

5.1 Fetch

1.apply redux-thunk middleware

```
import { applyMiddleware, createStore, compose } from 'redux';
import thunk from 'redux-thunk';
import createLogger from 'redux-logger';
import reducers from '../reducers';

var middlewares = compose(applyMiddleware(thunk), autoRehydrate(
));

export default function configureStore() {
  const store = createStore(reducers, undefined, middlewares);
  return store;
}
```

2.start & end action types

```
//todo action types
export const START_FETCH_ALL_TODOS = 'START_FETCH_ALL_TODOS';
export const FETCH_ALL_TODOS = 'FETCH_ALL_TODOS';
```

3.fetch flow

```
import * as types from '../constants/ActionTypes';
import * as APIs from '../constants/ServerAPIs';


function shouldFetchAllTodos(state) {
  const data = state.todos;
  if (data && data.isFetchingAllTodos) {
    return false
  }
  return true;
}

export function fetchAllTodos() {
  return async (dispatch, getState) =>{
    //verify
    if(!shouldFetchAllTodos(getState())){
      return Promise.resolve();
    }

    //dispatch fetch start action
    dispatch({type: types.START_FETCH_ALL_TODOS});

    //fetching
    const response = await fetch(APIs.allTodos);
    //response
    const data = await response.json();

    //dispatch fetch end action
    return dispatch({
      type: types.FETCH_ALL_TODOS,
      data
    });
  }
}
```

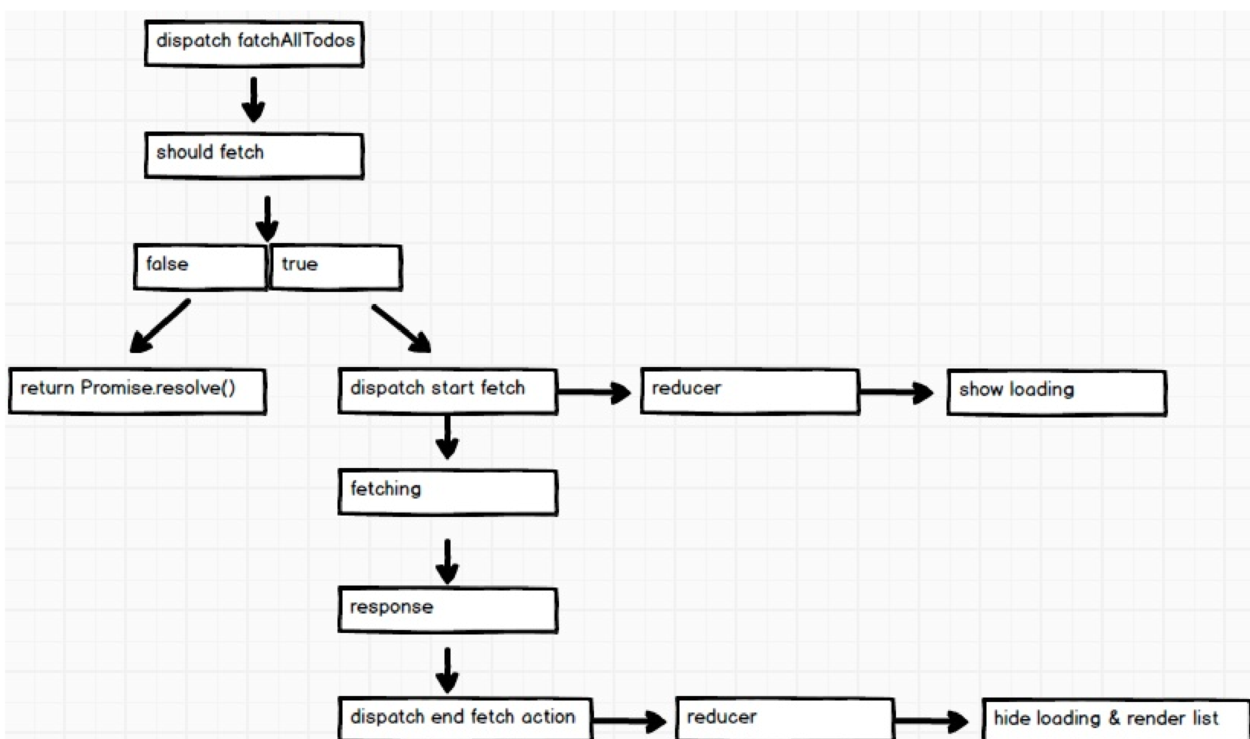
4.reducer

```
export default function todos(state = initialState, action) {
  switch (action.type) {
    case types.START_FETCH_ALL_TODOS:
      return Object.assign({}, state, {isFetchingAllTodos: true}
);

    case types.FETCH_ALL_TODOS:
      return Object.assign({}, state, {
        isFetchingAllTodos: false,
        data: action.data.data.reduce(function (pre, cur) {
          //remove duplicates
          !pre.find( key=> key.id===cur.id) && pre.push(cur);
          return pre;
        }, [...state.data])
      });
    ...
    ...
    default:
      return state
  }
}
```

5.dispatch & render

```
...
  componentDidMount(){
    //fetch data from server
    this.props.actions.fetchAllTodos();
  }
...
...
  renderLoading = () => {
    if (this.props.todos.isFetchingAllTodos) {
      return (
        <View style={styles.loading}>
          <Text style={styles.loadingText}>Loading...</Text>
        </View>
      )
    }
    return null;
  }
...
```



5.2 Persistent

1.AsyncStorage

2.apply redux-persist middlewear

```
import { AsyncStorage } from 'react-native';
import { applyMiddleware, createStore, compose } from 'redux';
import thunk from 'redux-thunk';
import {persistStore, autoRehydrate} from 'redux-persist';
import reducers from '../reducers';

var middlewares = compose(applyMiddleware(thunk), autoRehydrate(
));

export default function configureStore() {
  const store = createStore(reducers, undefined, middlewares);
  persistStore(store, {storage: AsyncStorage});
  return store;
}
```

5.3 Resources

- Redux Async Actions
- Todo React Native App

6 Router

6.1 Navigator

1.define routes

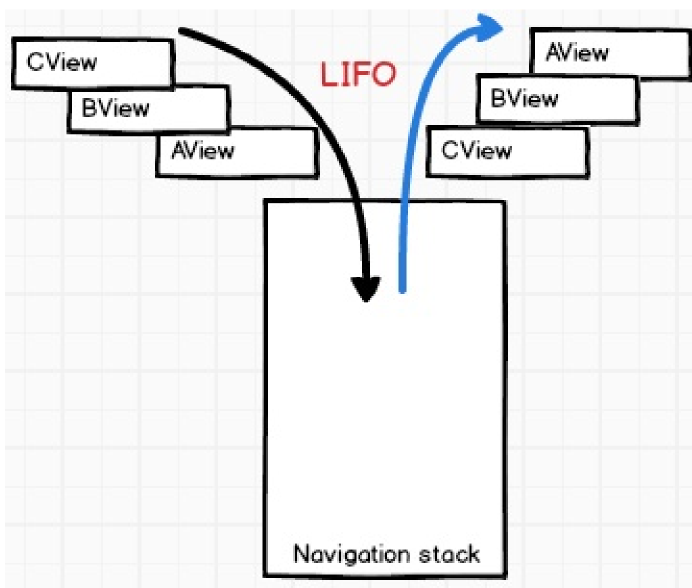
```
import MainTabsView from './MainTabsView';
import EditView from './EditView';
import BroswerView from './BroswerView';

const ROUTES = { MainTabsView,  BroswerView, EditView };
```

2.config Navigator

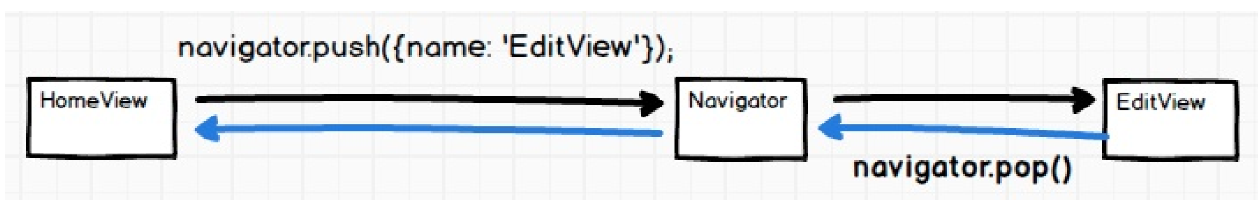
```
class App extends Component {
  renderScene = (route, navigator) => {
    let Scene = ROUTES[route.name];
    return <Scene {...route} navigator={navigator}/>;
  }
  configureScene = (route, routeStack) => {
    switch (route.name){
      case 'EditView':
        return Navigator.SceneConfigs.FloatFromBottom;
      default:
        return Navigator.SceneConfigs.PushFromRight;
    }
  }
  render() {
    return (
      <View style={styles.container}>
        <StatusBar barStyle="light-content"/>
        <Navigator
          initialRoute={{name: 'MainTabsView'}}
          renderScene={this.renderScene}
          configureScene={this.configureScene}/>
      </View>
    )
  }
}
```

3.forward & back




```
...
  handleEdit = ()=>{
    //Navigate forward to a new scene
    this.props.navigator.push({name: 'EditView', data: this.prop
s.data});
  }
...
```

```
...
  close = ()=>{
    //Transition back and unmount the current scene
    this.props.navigator.pop();
  }
...
```



4.onDidFocus & onWillFocus

```
...
componentDidMount(){
    this.currentRoute = this.props.navigator.navigationContext.c
urrentRoute;
    this.bindEvents();
  }
  componentWillUnmount(){
    this.unBindEvents();
  }
  bindEvents = ()=>{
    this.willFocusSubscription  = this.props.navigator.navigatio
nContext.addListener('willfocus', (event) => {
      if (this.currentRoute !== event.data.route) {
        this.setState({isVisible: false});
      }
    });
    this.didFocusSubscription  = this.props.navigator.navigation
Context.addListener('didfocus', (event) => {
      if (this.currentRoute === event.data.route) {
        this.setState({isVisible: true});
      }
    });
  }
  unBindEvents = ()=>{
    this.willFocusSubscription.remove();
    this.didFocusSubscription.remove();
  }
...
```

6.2 Resources

- Routing and Navigation in React Native

7 Native Modules

Somewhere in your RN codes, insert:

```
console.log(__fbBatchedBridge);
```

then in your debugger (http://localhost:8081/debugger-ui) you'll see something like below,

localhost:8081/debugger-ui

Elements Console Sources Network Timeline Profiles Resources Security Audits

React Native JS code runs inside this Chrome tab.

Press ⌘⌥J to open Developer Tools. Enable Pause On Caught Exceptions for a better debugging experience.

Status: Debugger session #10000 active.

JSONView Preserve log Show all messages

```
▼MessageQueue {_callableModules: Object, _queue: Array[4], _callbacks: Array[2], _callbackID: 2
 ▼RemoteModules: Object
   ▶AccessibilityManager: Object
   ▶ActionSheetManager: Object
   ▶AlertManager: Object
   ▶AppState: Object
   ▶AsyncLocalStorage: Object
   ▶Clipboard: Object
   ▶DemoManager: Object
   ▶DevLoadingView: Object
   ▶DevMenu: Object
   ▶ExceptionsManager: Object
   ▶ImageEditingManager: Object
   ▶ImageLoader: Object
   ▶ImageStoreManager: Object
   ▶ImageViewManager: Object
   ▶JSCExecutor: Object
   ▶KeyboardObserver: Object
   ▶LinkingManager: Object
   ▶LocationObserver: Object
   ▶NavigatorManager: Object
   ▶NetInfo: Object
   ▶Networking: Object
   ▶RedBox: Object
   ▶ScrollViewManager: Object
   ▶SettingsManager: Object
   ▶SourceCode: Object
   ▶StatusBarManager: Object
   ▶Timing: Object
   ▶UIManager: Object
   ▶Vibration: Object
   ▶ViewManager: Object
   ▶WebSocketModule: Object
   ▶WebViewManager: Object
   ▶__proto__: Object
  _callID: 204
 ▼_callableModules: Object
   ▶AppRegistry: Object
   ▶HMRClient: Object
   ▶HeapCapture: Object
   ▶JSTimersExecution: Object
   ▶PerformanceLogger: Object
   ▶RCTDebugComponentOwnership: Object
   ▶RCTDeviceEventEmitter: RCTDeviceEventEmitter
   ▶RCTEventEmitter: Object
   ▶RCTLog: function RCTLog()
   ▶RCTNativeAppEventEmitter: RCTDeviceEventEmitter
   ▶Systrace: Object
   ▶__proto__: Object
  _callbackID: 2
 ▶_callbacks: Array[2]
 ▶_debugInfo: Object
  _eventLoopStartTime: 1467083013666
  _lastFlush: 1467083013666
 ▶_queue: Array[4]
 ▶_remoteMethodTable: Object
 ▶_remoteModuleTable: Object
```

Custom Native Module

that's what we're going to talk about in this chapter.

7.1 iOS

7.2 Android

7.3 Resources

- An OpenGL App with React Native

8 Integration

Most of the time we are not starting a new app, we just want to use react-native to develop some new features, so integration should be a necessary skill for react-native developers.

Assume that you have create some features in the AwesomeProject, you want to create an exactly same environment for your current project.

```
~/rn_demo/AwesomeProject — -bash
Danychen:rn_demo danychen$ ls
AwesomeProject  MyAndroidApp    MyiOSApp
Danychen:rn_demo danychen$ cd AwesomeProject/
Danychen:AwesomeProject danychen$ react-native --version
react-native-cli: 0.2.0
react-native: 0.28.0
Danychen:AwesomeProject danychen$
```

Notice that version is very important after your app published. If you want to upgrade react-native and package codes, that almost means you don't want to maintain the old version any longer.

8.1 iOS

- ## Cocoapods with local path

Requirement : CocoaPods

After pod **version > 1.0**, you need to identify the target. Create 'Podfile' in project root folder :

```
target 'MyiOSApp' do
  pod 'React', :path => '../../AwesomeProject/node_modules/r
eact-native', :subspecs => [
      'Core',
      'RCTImage',
      'RCTNetwork',
      'RCTText',
      'RCTWebSocket',
  ]
end
```

then `pod install`

```
Danychen:MyiOSApp danychen$ pod install
Analyzing dependencies
Fetching podspec for `React` from `../../AwesomeProject/node_modules/react-nativ
e`
Downloading dependencies
Installing React (0.28.0)
Generating Pods project
Integrating client project

[!] Please close any current Xcode sessions and use `MyiOSApp.xcworkspace` for t
his project from now on.
Sending stats
Pod installation complete! There are 5 dependencies from the Podfile and 1 total
pod installed.
Danychen:MyiOSApp danychen$ pod --version
1.0.1
```

9.1.1 Package

```
react-native bundle
--platform ios
--dev false
--entry-file index.ios.js
--bundle-output ios/bundle/index.ios.bundle
--assets-dest ios/bundle
```

9.1.2 Image

- **Cache path**

8.2 Android

At first I followed the official instruction (which seems very simple) but lots of build or runtime error occurs 😂.

Such as:

`Can't find variable: __fbBatchedBridge (<unknown file>:1)`

`java.lang.UnsatisfiedLinkError: could find DSO to load: libreactnativejni.so`

`android.view.WindowManager$BadTokenException: Unable to add window android.view.ViewRootImpl$W@5d992cf -- permission denied for this window type`

But the demo works corrcetly, so I decided to copy the build settings of it. And finally it works normally on my Nexus 5X. Steps:

- Add the path to the root gradle file,

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.2'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
        maven {
            // All of React Native (JS, Obj-C sources, Android binaries) is installed from npm
            url "$projectDir/../AwesomeProject/node_modules/react-native/android"
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

- Modify the app gradle file,

```
apply plugin: 'com.android.application'
apply from: "../../AwesomeProject/node_modules/react-native/react.gradle"

def enableSeparateBuildPerCPUArchitecture = false        1

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.3"
    defaultConfig {
        applicationId "cc.danycoding.myandroidapp"
        minSdkVersion 16
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
        ndk {
            abiFilters "armeabi-v7a", "x86"
        }
    }

    splits {
        abi {
            reset()
            enable enableSeparateBuildPerCPUArchitecture
            universalApk false  // If true, also generate a universal APK
            include "armeabi-v7a", "x86"
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile "com.facebook.react:react-native:+"  // From node_modules
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.0.1'        2
}
```

*1. Official demo use this variable to control wheather building different apps for cpus, that will reduce the size of each app, I just ignore it here.

*2. The version support package matters.. As react-native Android use many open source projects, if you use some of them already, you should check the version or exclude the from dependencies. The list currently

```
compile 'com.android.support:appcompat-v7:23.0.1'
compile 'com.android.support:recyclerview-v7:23.0.1'
compile 'com.facebook.fresco:fresco:0.11.0'
compile 'com.facebook.fresco:imagepipeline-okhttp3:0.11.0'
compile 'com.fasterxml.jackson.core:jackson-core:2.2.3'
compile 'com.google.code.findbugs:jsr305:3.0.0'
compile 'com.squareup.okhttp3:okhttp:3.2.0'
compile 'com.squareup.okhttp3:okhttp-urlconnection:3.2.0'
compile 'com.squareup.okhttp3:okhttp-ws:3.2.0'
compile 'com.squareup.okio:okio:1.8.0'
compile 'org.webkit:android-jsc:r174650'
```

- Modify root gradle.properties,

MyAndroidApp × app × gradle-wrapper.properties × gradle.properties ×

```
# Project-wide Gradle settings.

# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.

# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html

# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
# Default value: -Xmx10248m -XX:MaxPermSize=256m
# org.gradle.jvmargs=-Xmx2048m -XX:MaxPermSize=512m -XX:+HeapDumpOnOutOfMemoryError -Dfile.encoding=UTF-8

# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. More details, visit
# http://www.gradle.org/docs/current/userguide/multi_project_builds.html#sec:decoupled_projects
# org.gradle.parallel=true
android.useDeprecatedNdk=true
```

- Add proguard rules,

idApp × app × proguard-rules.pro × gradle-wrapper.properties × gradle.properties × settings.gradle × local.properties ×

```
# Keep our interfaces so they can be used by other ProGuard rules.
# See http://sourceforge.net/p/proguard/bugs/466/
-keep,allowobfuscation @interface com.facebook.proguard.annotations.DoNotStrip
-keep,allowobfuscation @interface com.facebook.proguard.annotations.KeepGettersAndSetters

# Do not strip any method/class that is annotated with @DoNotStrip
-keep @com.facebook.proguard.annotations.DoNotStrip class *
-keepclassmembers class * {
    @com.facebook.proguard.annotations.DoNotStrip *;
}

-keepclassmembers @com.facebook.proguard.annotations.KeepGettersAndSetters class * {
  void set*(***);
  *** get*();
}

-keep class * extends com.facebook.react.bridge.JavaScriptModule { *; }
-keep class * extends com.facebook.react.bridge.NativeModule { *; }
-keepclassmembers,includedescriptorclasses class * { native <methods>; }
-keepclassmembers class *  { @com.facebook.react.uimanager.UIProp <fields>; }
-keepclassmembers class *  { @com.facebook.react.uimanager.annotations.ReactProp <methods>; }
-keepclassmembers class *  { @com.facebook.react.uimanager.annotations.ReactPropGroup <methods>; }

-dontwarn com.facebook.react.**

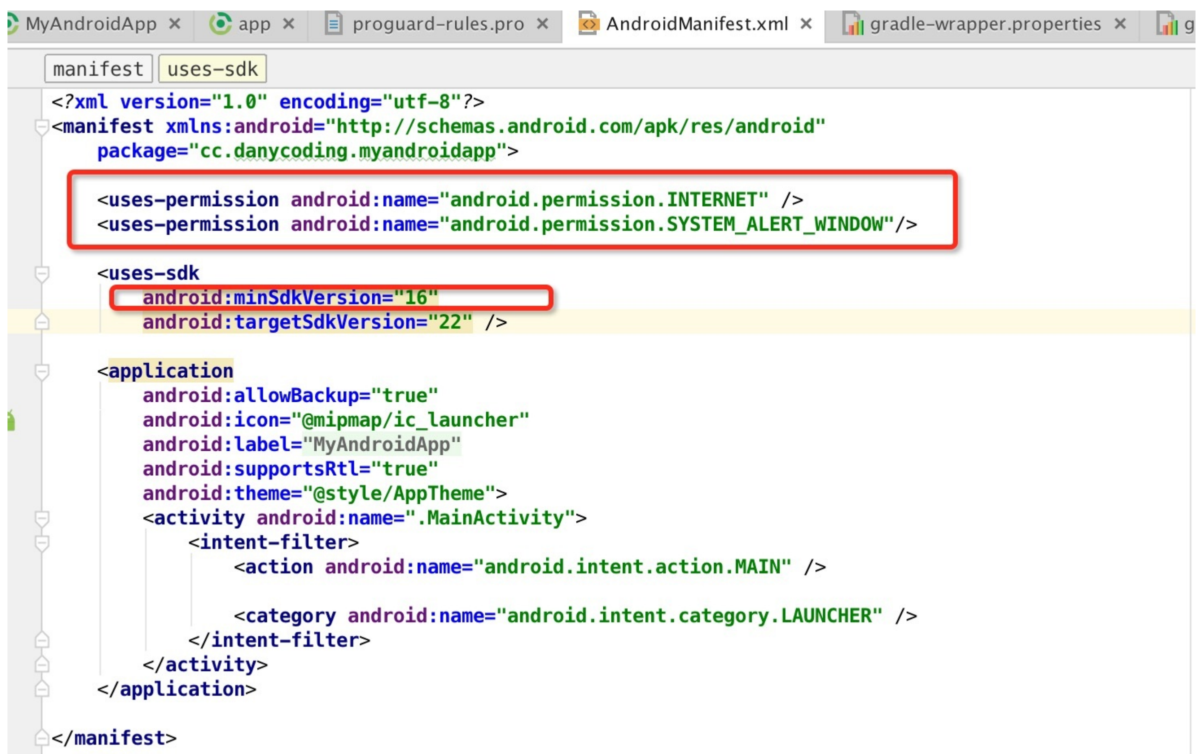
# okhttp

-keepattributes Signature
-keepattributes *Annotation*
-keep class okhttp3.** { *; }
-keep interface okhttp3.** { *; }
-dontwarn okhttp3.**

# okio

-keep class sun.misc.Unsafe { *; }
-dontwarn java.nio.file.*
-dontwarn org.codehaus.mojo.animal_sniffer.IgnoreJRERequirement
-dontwarn okio.**
```

- AndroidManifest.xml, you can remove the permission if you don't need debug mode.



9.2.1 Package

```
react-native bundle
--platform android
--dev false
--entry-file index.android.js
--bundle-output android/bundle/index.android.bundle
--assets-dest android/bundle/
```

8.3 Before publishing

- **Turn off debug Settings.**
- **Implementation of exception handler.**

8.3 Resources

- Integrating with Existing Apps - iOS
- Integrating with Existing Apps - Android

9.3 Resources

- React Native Module for CodePush
- JSPatch

10 Performance (draft)

10.1 shouldComponentUpdate

This chapter can be applied to all react apps.

shouldComponentUpdate

React is usually fast, but you still can improve performance by optimizing function shouldComponentUpdate. By default it returns true, if returns false, the render function will be skipped.

This function is frequently invoked when states or props are changed. So it's important to **keep it simple and fast**. When you called `setState` , the `render` function will always be excuted even if previous states are equal to current. This is where we can make some optimization.

demo1

In demo1, when click button, it will set same state, but render times will still increase.

demo2

In demo2, we check the value of name is equal to before or not, if equal return false, then we reduce the times of render function.

But if our states structure is complicated, such as `{ a: { b: { c: [1, 2, 3] }}}` , we have to compare them deeply. This is obviously against the rules we mentioned above, **keep shouldComponentUpdate simple**

Immutable-js

Immutable is a concept from functional programming, one of immutable data features is that it can not be modified after being created. So there are some algorithm to create hash for every data structure(for more detail). We can use this feature to prevent deeply compare, shallow compare is enough. Here we will use immutable-js from facebook

demo3

In demo3, we click first button several times, times will only plus one time, click second button , times will increase.

Resources

- ReactJS
- React Native
- awesome-react
- awesome-react-native
- build with react

Books

- Learning React Native
- Developing a React Edge